Altova RaptorXML Server 2025

Benutzer- und Referenzhandbuch

Altova RaptorXML Server 2025
Benutzer- und Referenzhandbuch

Published: 2025

Inhaltsverzeichnis

1 Einführung

Altova **RaptorXML Server** (in der Folge als RaptorXML bezeichnet) ist Altovas ultraschneller XML- und XBRL*-Prozessor der dritten Generation, der für die neuesten Standards und parallele Rechnerumgebungen optimiert wurde. RaptorXML lässt sich plattformübergreifend einsetzen und ermöglicht dank der Nutzung moderner Multi-Core Computer die ultraschnelle Verarbeitung von XML- und XBRL-Daten.

Anmerkung: Die XBRL-Verarbeitung steht nur in RaptorXML+XBRL Server, nicht aber in RaptorXML Server zur Verfügung.

Diese Dokumentation

Diese Dokumentation ist im Lieferumfang der Applikation enthalten und steht auch auf der Altova Website zur Verfügung. Diese Dokumentation ist in die folgenden Abschnitte gegliedert:

Altova Website: Server für die XML-Validierung, XML-Validierung

Letzte Aktualisierung: 17.03.2025

2 Informationen zu RaptorXML Server

Editionen und Betriebssysteme

RaptorXML steht in Form von zwei verschiedenen Editionen zur Verfügung. Jede davon eignet sich für unterschiedliche Zwecke. Diese Editionen sind im Abschnitt Editionen und Schnittstellen 11 beschrieben. RaptorXML steht für Windows, Linux und macOS zur Verfügung. Nähere Informationen zu unterstützten Systemen finden Sie im Abschnitt Systemanforderungen 15.

Funktionalitäten und unterstützte Spezifikationen

RaptorXML unterstützt die Validierung von XML-Dateien, die XSLT-Transformation und XQuery-Ausführung. Für jede dieser Aufgaben stehen zahlreiche Optionen zur Verfügung. Im Abschnitt Funktionalitäten 16 finden Sie eine umfangreiche Liste wichtiger Schlüsselfunktionen. Der Abschnitt Unterstützte Spezifikationen 18 enthält eine detaillierte Liste der Spezifikationen, die RaptorXML erfüllt. Nähere Informationen finden Sie auf der RaptorXML-Seite der Altova-Website.

2.1 Editionen und Schnittstellen

Editionen

RaptorXML steht in Form der folgenden Editionen zur Verfügung:

- *RaptorXML Server* ist ein schneller serverbasierter Prozessor zur Validierung und flexiblen Verarbeitung von XML-, XML-Schema-, XML-Signatur-, XSLT- und XQuery-Dokumenten.
- *RaptorXML+XBRL Server* bietet alle Funktionen von RaptorXML Server sowie eine breite Palette an Funktionen zur Verarbeitung von XBRL-Dokumenten.

Eine Liste der unterstützten Spezifikationen[18] finden Sie hier[18].

Schnittstellen

Nach Installation von RaptorXML kann das Produkte auf eine der folgenden Arten aufgerufen werden:

- *Befehlszeilenschnittstelle* (CLI): verfügbar für Windows, Linux und macOS
- *HTTP REST-Client-Schnittstelle*: verwendet die HTTP-Schnittstelle von RaptorXML.
- *COM/.NET-Server-Schnittstelle (Windows):* verwendet (i) die COM/.NET API und die (ii) HTTP-Schnittstelle von RaptorXML.
- *Java-Schnittstelle* (*Windows, Linux, macOS*): verwendet (i) die Java API und die (ii) HTTP REST-Schnittstelle von RaptorXML
- *Altova XMLSpy-Schnittstelle:* RaptorXML kann von der Benutzeroberfläche von Altova XMLSpy aus aufgerufen werden.
- *Python-Prozessorschnittstelle* verwendet (i) in Ihrer Python-Umgebung ein RaptorXML. Python-Wheel und (ii) in Ihrem Python-Skript die Python API von RaptorXML. Auf diese Art können RaptorXML-Funktionalitäten in Python-Skripts zusammen mit Python-Paketen von Drittanbietern verwendet werden.
- *.NET-Prozessorschnittstelle:* verwendet (i) eine RaptorXML DLL und (ii) die .NET API von RaptorXML, um Ihre eigenen Applikationen, die RaptorXML-Funktionalitäten verwenden, zu erstellen.

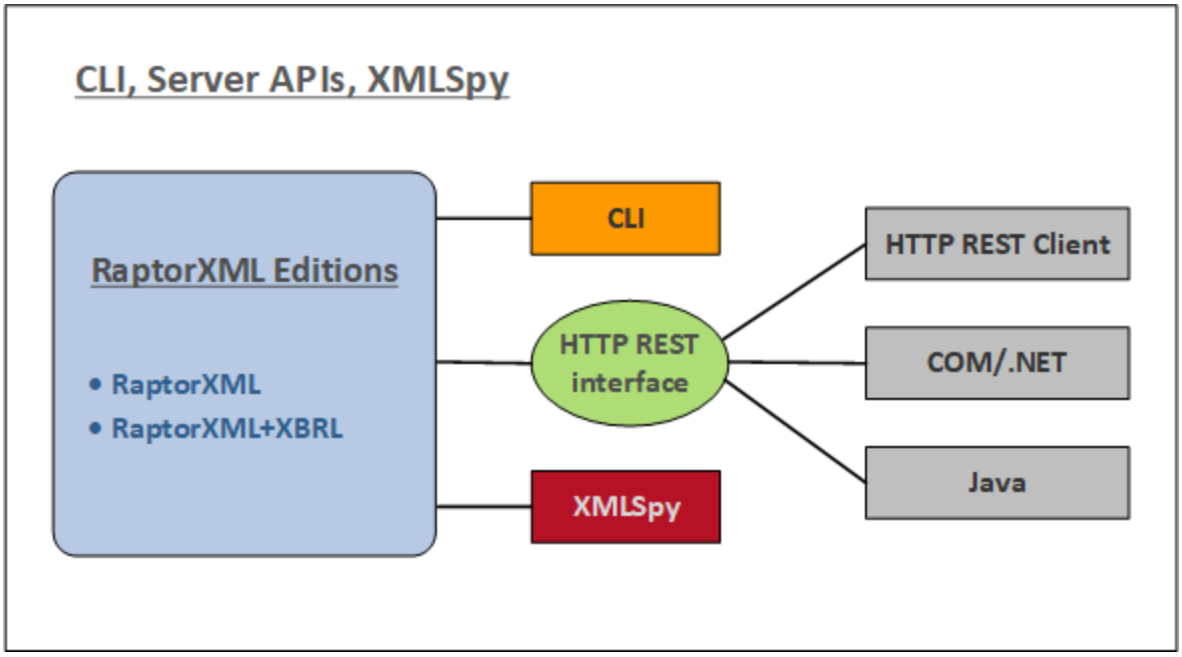
Diese sieben Schnittstellen können in vier Gruppen unterteilt werden:

- Befehlszeilenschnittstelle(CLI)[58]
- Server APIs: HTTP, COM/.NET, Java[266]
- Prozessor-APIs: Python und .NET[387]
- Altova XMLSpy[13]

CLI, Server APIs und Altova XMLSpy

Der Aufruf über die CLI, die Server APIs und Altova XMLSpy ist im nachstehenden Diagramm dargestellt.

In RaptorXML Server ist eine HTTP REST-Schnittstelle definiert, über die Clients Validierungsaufträge an den Server weitergeben können. Clients können die HTTP REST-Schnittstelle entweder direkt oder über die oberste Ebene der COM/.NET- und Java Server API aufrufen. Diese APIs bieten einfach zu verwendende COM/.NET- und Java-Klassen zur Verwaltung der Erstellung und Absendung der HTTP REST Requests. Zusätzlich dazu kann Altova XMLSpy so konfiguriert werden, dass Validierungsaufträge über einen entfernten RaptorXML Server ausgeführt werden.



Befehlszeilenschnittstelle (CLI)

- RaptorXML ist auf dem Rechner, auf dem er installiert ist, lizenziert und diese Lizenz wird über die Befehlszeile aufgerufen.
- Kann unter Windows, Linux und macOS installiert werden.
- Ermöglicht die Validierung und Verarbeitung von XML-, XML-Schema-, XML-Signatur-, XQuery- und XSLT-Dokumenten über die Befehlszeile[58].
- Python 3.11.8 ist in RaptorXML gebündelt und wird bei Aufruf eines Python-Skript mit der Option `--script` verwendet.

HTTP REST-Client-Schnittstelle

- RaptorXML ist auf dem Rechner, auf dem er installiert ist, lizenziert und diese Lizenz wird über eine HTTP REST-Client-Schnittstelle[268] aufgerufen.
- Client-Requests werden im JSON-Format gesendet. Jedem Request wird ein Auftragsverzeichnis auf dem Server zugewiesen, in dem die Ausgabedateien gespeichert werden. Die Server-Antworten an den Client enthalten alle relevanten Informationen zum Auftrag.
- Python 3.11.8 ist in RaptorXML gebündelt und wird bei Aufruf eines Python-Skript mit der Option `--script` verwendet.

COM/.NET-Schnittstelle

- Steht nur unter Windows zur Verfügung.
- RaptorXML wird bei der Installation automatisch als COM-Server-Objekt registriert und kann daher von Applikationen und Skriptingsprachen aus, die Programmierunterstützung für COM-Aufrufe haben, aufgerufen werden.
- RaptorXML ist auf dem Rechner, auf dem er installiert ist, lizenziert.

- Die .NET-Schnittstelle ist als Wrapper rund um die COM-Schnittstelle implementiert.
- Die COM/.NET Server API[311] von RaptorXML stellt Objekte bereit, mit Hilfe derer RaptorXML-Funktionalitäten in COM/.NET-Skriptsprachen aufgerufen werden können.
- Python 3.11.8ist in RaptorXML gebündelt und wird bei Aufruf eines Python-Skript mit der Option `--script` verwendet.

Java-Schnittstelle

- RaptorXML ist auf dem Rechner, auf dem er installiert ist, lizenziert und diese Instanz wird über ein Java-Programm aufgerufen.
- RaptorXML-Funktionalitäten stehen in der Java Server API[320] als Java-Klassen zur Verfügung, die in Java-Programmen verwendet werden können.
- Python 3.11.8ist in RaptorXML gebündelt und wird bei Aufruf eines Python-Skript mit der Option `--script` verwendet.

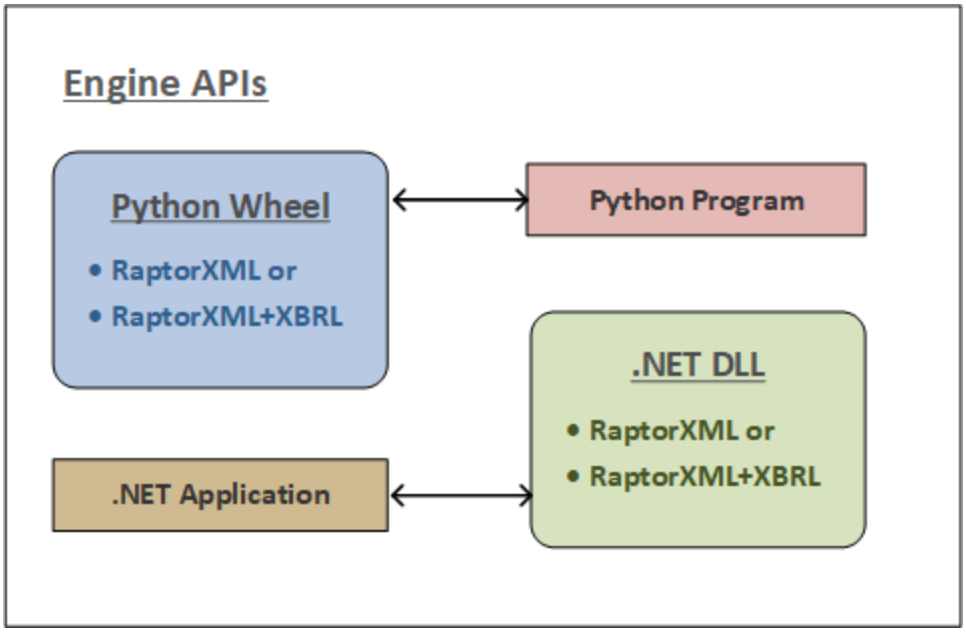
Altova XMLSpy

- Wenn Sie Altova XMLSpy installiert und lizenziert haben und wenn XMLSpy RaptorXML Server über ein Netzwerk aufrufen kann, können Sie von der Benutzeroberfläche von XMLSpy aus RaptorXML Server aufrufen, um damit XML-Dokumente zu validieren und XSLT- und XQuery-Transformationen auszuführen.
- Sie können das aktive oder alle Dokumente in einem XMLSpy-Projektordner validieren.
- Die Validierungsergebnisse werden im Fenster "Meldungen" der XMLSpy-Benutzeroberfläche angezeigt.
- Sie können in XMLSpy über die Prozessoren in XMLSpy oder RaptorXML Server (i) Dokumente validieren oder (ii) XSLT/XQuery-Transformationen ausführen.
- Einer der Hauptvorteile der Verwendung von Raptor ist, dass Sie Validierungen mit Hilfe einer breiten Palette an Validierungsoptionen genau konfigurieren können. Außerdem können Sie in XMLSpy eine Gruppe von Raptor-Optionen als "Konfiguration" speichern und dann eine der von Ihnen definierten Konfigurationen für eine bestimmte Raptor-Validierung auswählen. Auch bei der Validierung von großen Datenmengen erweist sich die Verwendung von Raptor als vorteilhaft.

Prozessor-APIs

Die Prozessor-APIs[387] unterscheiden sich insofern von den Server-APIs, als RaptorXML im Python Wheel und in den .NET DLL enthalten ist, die von Python- bzw. .NET-Applikationen verwendet werden. (*siehe Abbildung unten*). Diese Programme/Applikationen können die RaptorXML-Funktionalitäten nur über die Python API[391] bzw. die .NET API[401] von Raptor aufrufen.

Anmerkung: Der über die Python API[391] und die .NET API[401] verfügbare Funktionsumfang ist erheblich größer als der über die CLI oder die Server APIs bereitgestellte Funktionsumfang. Ein Beispiel dafür sind die Funktionen zum Auslesen und Bearbeiten von Daten.



Python-Schnittstelle

- RaptorXML steht in einem Python Wheel-Paket zur Verfügung, das in Ihrer 3.11.8-Umgebung installiert werden kann.
- Es kann ein Python-Programm geschrieben werden, das Objekte aus der Python API[391] von RaptorXML verwendet. Über diese API stehen viel mehr Funktionalitäten als über die CLI zur Verfügung. Außerdem können diese mit den Funktionalitäten von Drittanbieter-Bibliotheken in Ihrer Python-Umgebung kombiniert werden.
- Bei Aufruf von RaptorXML-Funktionalitäten über das Python Wheel von RaptorXML, wird vor Ausführung des Befehls überprüft, ob auf diesem Rechner eine gültige RaptorXML-Lizenz zur Verfügung steht.

.NET-Schnittstelle

- RaptorXML steht in einer DLL zur Verfügung, die in eine Applikation, die das .NET Framework unterstützt, eingebettet werden kann. Nähere Informationen zur API finden Sie im Abschnitt .NET Framework API[401].
- Über die .NET API[401] von RaptorXML erhalten Sie Zugriff auf RaptorXML. Über diese API stehen viel mehr Funktionalitäten als über die CLI von RaptorXML zur Verfügung.
- Bei Aufruf von RaptorXML-Funktionalitäten über eine .NET-Applikation wird vor Ausführung des Befehls überprüft, ob auf diesem Rechner eine gültige RaptorXML-Lizenz zur Verfügung steht.

2.2 Systemanforderungen

RaptorXML Server läuft auf den folgenden Betriebssystemen:

Windows

- Windows 10, Windows 11
- Windows Server 2016 oder höher

Linux

- Red Hat Enterprise Linux 7 oder neuer
- CentOS 7, CentOS Stream 8
- Debian 10 oder neuer
- Ubuntu 20.04, 22.04, 24.04
- AlmaLinux 9.0
- Rocky Linux 9.0

Voraussetzungen

- Führen Sie die Installation entweder als **root**-Benutzer durch oder als Benutzer mit **sudo**-Rechten.
- Die vorherige Version von RaptorXML Server muss deinstalliert werden, bevor Sie eine neue Version installieren.
- Wenn Sie beabsichtigen, die Diagrammfunktionalität von Altova zu verwenden, muss auf Ihrem System mindestens eine Schriftart installiert sein, damit die Diagramme korrekt dargestellt werden können. Installierte Schriftarten können Sie z.B. mit dem Befehl **`fc-list`** aus der Fontconfig-Bibliothek auflisten.
- Um die Applikation installieren und ausführen zu können, werden die folgenden Bibliotheken benötigt. Falls die unten angeführten Pakete auf Ihrem Linux-Rechner noch nicht zur Verfügung stehen, führen Sie die Befehl `yum` (oder ggf. `apt-get`) aus, um sie zu installieren.

CentOS, RedHat	Debian	Ubuntu
krb5-libs	libgssapi-krb5-2	libgssapi-krb5-2

macOS

- macOS 12 oder neuer

RaptorXML steht sowohl für 32-Bit- als auch für 64-Bit-Rechner zur Verfügung. Dabei handelt es sich um x86- und amd64- (x86-64) Instruction-Set-basierte Kerne: Intel Core i5, i7, XEON E5. Um RaptorXML über eine COM-Schnittstelle verwenden zu können, muss der Benutzer Rechte zur Benutzung der COM-Schnittstelle haben, damit er die Applikation registrieren kann und die entsprechenden Applikationen und/oder Skripts ausführen kann.

2.3 Funktionalitäten

RaptorXML enthält die unten aufgelisteten Funktionalitäten. Die meisten Funktionalitäten können sowohl über die Befehlszeilenschnittstelle als auch die CO-Schnittstelle verwendet werden. Ein bedeutender Unterschied ist, dass bei Verwendung der CO-Schnittstelle unter Windows Dokumente über die Applikation oder Skript-Code anhand von Textstrings konstruiert werden können (anstatt XML-, DTD-, XML-Schema-, XSLT- oder XQuery-Dateien zu referenzieren).

XML-Validierung

- Validierung des bereitgestellten XML-Dokuments anhand einer internen oder externen DTD oder eines internen oder externen Schemas.
- Überprüfung der Wohlgeformtheit von XML-, DTD-, XML-Schema, XSLT- und XQuery-Dokumenten.

XSLT-Transformationen

- Transformierung von XML-Dateien anhand des bereitgestellten XSLT 1.0-, 2.0- oder 3.0-Dokuments
- XML- und XSLT-Dokumente können als Datei (über eine URL) oder bei Verwendung der COM-Schnittstelle als Textstring bereitgestellt werden.
- Rückgabe der Ausgabe als Datei (in einem definierten Ordner) oder bei Verwendung der COM-Schnittstelle als Textstring
- XSLT-Parameter können über die Befehlszeile und über die COM-Schnittstelle geliefert werden.
- Spezialisierte Verarbeitung dank Altova- sowie XBRL-, Java- und .NET-Erweiterungsfunktionen. Dies ermöglicht z.B. die Erstellung von Diagrammen und Barcodes in Ausgabedokumenten.

XQuery-Ausführung

- Ausführung von XQuery 1.0- und 3.0-Dokumenten
- XQuery- und XML-Dokumente können als Datei (über eine URL) oder bei Verwendung der COM-Schnittstelle als Textstring bereitgestellt werden.
- Rückgabe der Ausgabe als Datei (in einem definierten Ordner) oder bei Verwendung der COM-Schnittstelle als Textstring
- Bereitstellung externer XQuery-Variablen über die Befehlszeile und die COM-Schnittstelle
- Inkludiert in den Serialisierungsoptionen sind: Ausgabekodierung, Ausgabemethode (ob in XML, XHTML, HTML oder Text), Weglassen der XML-Deklaration und Einrückung.

JSON- und Avro-Validierung/Konvertierung

- Validierung von JSON-Schema- und Avro-Schema-Dokumenten
- Validierung von JSON-Instanzen anhand von JSON-Schemas und Avro-Schemas
- Validierung von Avro-Binärdateien
- Konvertierung von Avro-Binärdateien in Avro-Schemas und Avro-Daten im JSON-Format
- Konvertierung von Avro-JSON-Daten in Avro-Binärdateien

Hochleistungsfunktionen

- Extrem hohe Verarbeitungsgeschwindigkeit dank optimiertem Code
  - Native Instruction-Set Implementierungen
  - 32-Bit- oder 64-Bit-Version
- Extrem niedriger Arbeitsspeicherbedarf
  - extrem kompakte speicherresidente Darstellung des XML Information Set
  - Streaming der Instanzvalidierung
- Plattformübergreifende Funktionalitäten
- Hochgradig skalierbarer Code für Multi-CPU/Multi-Core/Parallel Computing
- Paralleles Laden, Validierung und Verarbeiten entsprechend dem Design

Funktionen für Entwickler

- Ausgezeichnete Fehlerberichte
- Windows Server-Modus und Unix Daemon-Modus (über Befehlszeilenoptionen)
- Inkludierter Python 3.x Interpreter für das Skripting
- Dank der RaptorXML-Funktionalität in einem Python-Paket können die Funktionalitäten als Python-Bibliothek importiert werden.
- .NET Framework API bietet Zugriff auf das zugrunde liegende XML-Datenmodell
- COM API auf Windows-Betriebssystemen
- Java API auf allen Systemen
- XPath-Erweiterungsfunktionen Java, .NET und mehr
- Streaming-Serialisierung
- Integrierter HTTP-Server mit REST-Validierungs-API

Nähere Informationen dazu finden Sie im Abschnitt Unterstützte Spezifikationen 18 und auf der Altova Website.

2.4 Unterstützte Spezifikationen

RaptorXML unterstützt die folgenden unten aufgelisteten Spezifikationen.

W3C Recommendations

Website: World Wide Web Consortium (W3C)

- Extensible Markup Language (XML) 1.0 (Fifth Edition)
- Extensible Markup Language (XML) 1.1 (Second Edition)
- Namespaces in XML 1.0 (Third Edition)
- Namespaces in XML 1.1 (Second Edition)
- XML Information Set (Second Edition)
- XML Base (Second Edition)
- XML Inclusions (XInclude) Version 1.0 (Second Edition)
- XML Linking Language (XLink) Version 1.0
- XML Schema Part 1: Structures Second Edition
- XML Schema Part 2: Datatypes Second Edition
- W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures
- W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes
- XPointer Framework
- XPointer xmlns() Scheme
- XPointer element() Scheme
- XML Path Language (XPath) Version 1.0
- XSL Transformations (XSLT) Version 1.0
- XML Path Language (XPath) 2.0 (Second Edition)
- XSL Transformations (XSLT) Version 2.0
- XQuery 1.0: An XML Query Language (Second Edition)
- XQuery 1.0 and XPath 2.0 Functions and Operators (Second Edition)
- XSLT 2.0 and XQuery 1.0 Serialization (Second Edition)
- XML Path Language (XPath) 3.0
- XML Path Language (XPath) 3.1
- XQuery 3.0: An XML Query Language
- XQuery Update Facility 1.0
- XPath and XQuery Functions and Operators 3.0
- XSLT and XQuery Serialization 3.0

W3C Working Drafts & Candidate Recommendations

Website: World Wide Web Consortium (W3C)

- XSL Transformations (XSLT) Version 3.0 (Untergruppe)
- XQuery 3.1: An XML Query Language
- XPath and XQuery Functions and Operators 3.1
- XQuery Update Facility 3.0
- XSLT and XQuery Serialization 3.1

OASIS-Standards

Website: OASIS-Standards

- XML Catalogs V 1.1 - OASIS Standard V1.1

JSON/Avro-Standards

Websites: JSON Schema und Apache Avro

- JSON Schema Draft 4
- JSON Schema Draft 6
- JSON Schema Draft 7
- JSON Schema Draft 2019-09
- JSON Schema Draft 2020-12
- Apache Avro™ 1.8.1

2.5 Wichtige Änderungen

Unten finden Sie Änderungen in den einzelnen Versionen, die Sie eventuell beachten sollten.

v2024

Die Befehlszeilenoption **`--network-timeout`** erhält ab dieser Release einen Wert in Millisekunden (und nicht wie in früheren Versionen in Sekunden). Die Option kann für eine Reihe von Befehlen definiert werden und wird unter der Beschreibung des Befehls unter *Allgemeine Optionen* aufgelistet. Ein Beispiel dazu finden Sie unter dem Befehl valxml-withxsd (xsi) 65.

3 Installation und Lizenzierung

In diesem Abschnitt finden Sie Anleitungen zum Installieren, Lizenzieren und Konfigurieren. Es ist in die folgenden Abschnitte gegliedert:

3.1 Einrichten unter Windows

In diesem Abschnitt werden die Installation[22] und Lizenzierung von RaptorXML Server auf Windows-Systemen beschrieben. Bei der Einrichtung müssen die folgenden Schritte durchgeführt werden:

1. Installation von RaptorXML Server[22]
2. Installation von LicenseServer[27]
3. Starten von LicenseServer und RaptorXML Server[28]
4. Registrieren von RaptorXML Server auf LicenseServer[30]
5. Zuweisen einer Lizenz zu RaptorXML Server[31]

Die einzelnen oben beschriebenen Schritte müssen nicht in genau derselben Reihenfolge wie aufgelistet erfolgen. Bevor Sie beginnen, muss jedoch zunächst die Installation durchgeführt werden. Außerdem muss RaptorXML Server auf LicenseServer registriert werden, bevor Sie RaptorXML Server über LicenseServer eine Lizenz zuweisen können.

Systemvoraussetzungen (Windows)

Beachten Sie die folgenden Systemvoraussetzungen:

- Windows 10, Windows 11
- Windows Server 2016 oder höher

Voraussetzungen

Beachten Sie dabei die folgenden Voraussetzungen:

- Führen Sie die Installation als Benutzer mit Admin-Rechten durch.
- Ab Version 2021 kann eine 32-Bit-Version von RaptorXML Server nicht über eine 64-Bit-Version oder eine 64-Bit-Version nicht über eine 32-Bit-Version installiert werden. Sie müssen (i) die ältere Version entweder vor der Installation der neueren Version entfernen oder (ii) auf eine neuere Version, die dieselbe Bit-Version wie Ihre ältere Version hat, aktualisieren.

3.1.1 Installation unter Windows

Installation von RaptorXML Server

RaptorXML Server kann auf Windows-Systemen folgendermaßen installiert werden:

- Als separates eigenständiges Server-Produkt. Um RaptorXML Server zu installieren, laden Sie das RaptorXML Server-Installationsprogramm herunter und starten Sie es. Befolgen Sie die Anweisungen auf dem Bildschirm.
- Um RaptorXML Server im Rahmen des FlowForce Server-Pakets zu installieren, laden Sie das FlowForce Server-Installationsprogramm herunter und starten Sie es. Befolgen Sie die Anweisungen auf dem Bildschirm und stellen Sie sicher, dass Sie die Option zur Installation von RaptorXML Server aktiviert haben.

Sowohl das Installationsprogramm von RaptorXML Server als auch das von FlowForce Server steht im Altova Download Center (https://www.altova.com/de/download.html) zur Verfügung. Sie können im Feld links unten im

Assistenten die Installationssprache auswählen. Beachten Sie, dass Sie damit auch die Standardsprache von RaptorXML Server definieren. Sie können die Sprache später über die Befehlszeile ändern.

Nach der Installation befindet sich die ausführbare RaptorXML Server-Datei standardmäßig unter dem folgenden Pfad:

```
<ProgramFilesFolder>\Altova\RaptorXMLServer2025\bin\RaptorXML.exe
```

Alle erforderlichen Registrierungen zur Verwendung von RaptorXML Server über eine COM-Schnittstelle, als Java-Schnittstelle und in der :NET-Umgebung werden vom Installationsprogramm vorgenommen.Dazu gehört auch die Registrierung der ausführbaren RaptorXML Server-Datei als COM-Server-Objekt und das Hinzufügen der `Altova.RaptorXML.dll`-Datei zur .NET-Referenzbibliothek.

Deinstallieren von RaptorXML Server

Deinstallieren Sie RaptorXML Server folgendermaßen:

1. Klicken Sie mit der rechten Maustaste auf die Windows-Schaltfläche **Start** und wählen Sie **Einstellungen**.
2. Öffnen Sie die Systemsteuerung (Geben Sie die ersten Buchstaben davon ein und klicken Sie auf den Vorschlag).
3. Klicken Sie unter *Programme* auf **Programm deinstallieren**.
4. Wählen Sie in der Systemsteuerung RaptorXML Server aus und klicken Sie auf **Deinstallieren**.

Testlizenz

Sie haben während der Installation die Option, eine 30-Tage-Testlizenz für RaptorXML Server anzufordern. Die Testlizenz wird nach Erhalt der Anforderung an die von Ihnen registrierte E-Mail-Adresse gesendet.

3.1.2 Installation auf Windows Server Core

Windows Server Core ist eine Windows Minimalinstallation, bei der eine Reihe von Funktionen der Benutzeroberfläche nicht verwendet wird. Sie können RaptorXML Server folgendermaßen auf einem Windows Server Core-Rechner installieren:

1. Laden Sie die ausführbare RaptorXML Server-Installationsdatei von der Altova Website herunter. Diese Datei hat den Namen **`RaptorXMLServer<version>.exe`**. Stellen Sie sicher, dass die ausführbare Datei zu Ihrer Server-Plattform (32-Bit oder 64-Bit) passt.
2. Führen Sie auf einem Standard-Windows-Rechner (und nicht dem Windows Server Core-Rechner) den folgenden Befehl aus: **`RaptorXMLServer<version>.exe /u`**. Dadurch wird die **`.msi`**-Datei im selben Ordner entpackt, in dem sich auch die ausführbare Installationsdatei befindet.
3. Kopieren Sie die entpackte **`.msi`**-Datei auf den Windows Server Core-Rechner.
4. Wenn Sie eine frühere Version von RaptorXML Server aktualisieren, beenden Sie zuerst RaptorXML Server, bevor Sie den nächsten Schritt durchführen.
5. Verwenden Sie die **`.msi`**-Datei für die Installation, indem Sie den Befehl **`msiexec /i RaptorXMLServer.msi`** ausführen. Dadurch wird die Installation auf Windows Server Core gestartet.

Anmerkung: Bei Installation eines Upgrade auf eine Hauptversion können Sie Ihre RaptorXML Server-Einstellungen mit Hilfe der in den Unterabschnitten dieses Abschnitts aufgelisteten Eigenschaften beibehalten: (i) Webserver-Eigenschaften 25, (ii) SSL-Webserver-Eigenschaften 25 und (iii) Diensteigenschaften 26.

Achtung: Bewahren Sie die .msi-Datei auf!

Beachten Sie die folgenden Punkte:

- Speichern Sie die extrahierte **.msi**-Datei an einem sicheren Ort. Sie benötigen diese später, um das Produkt zu deinstallieren, zu reparieren oder die Installation anzupassen.
- Wenn Sie die MSI-Datei umbenennen möchten, tun Sie das, bevor Sie RaptorXML Server installieren.
- Der Name der MSI-Datei wird in der Registry gespeichert. Sie können den Dateinamen dort aktualisieren, falls er geändert wurde.

Registrieren von RaptorXML Server auf LicenseServer

Wenn Sie RaptorXML Server zum ersten Mal installieren oder ein Upgrade auf eine **Hauptversion** installieren, müssen Sie RaptorXML Server auf einem Altova LicenseServer in Ihrem Netzwerk registrieren. Wenn Sie ein Upgrade auf eine Nicht-Hauptversion von RaptorXML Server installieren, kennt das Installationsprogramm die vorherige LicenseServer-Registrierung, daher muss RaptorXML Server nicht auf dem LicenseServer registriert werden. Wenn Sie den von RaptorXML Server verwendeten LicenseServer jedoch zu irgendeinem Zeitpunkt wechseln möchten, müssen Sie RaptorXML Server auf dem neuen LicenseServer registrieren.

Um RaptorXML Server bei der Installation auf einem Altova LicenseServer zu registrieren, führen Sie den Installationsbefehl mit der Eigenschaft **REGISTER_WITH-LICENSE_SERVER** aus, wie unten aufgelistet, und geben Sie den Namen oder die Adresse des LicenseServer-Rechners als Wert der Eigenschaft an, z.B:

```
msiexec /i RaptorXMLServer.msi REGISTER_WITH_LICENSE_SERVER="localhost"
```

Um RaptorXML Server nach der Installation auf einem Altova LicenseServer zu registrieren, starten Sie den folgenden Befehl:

```
msiexec /r RaptorXMLServer.msi REGISTER_WITH_LICENSE_SERVER="<MyLS-IPAddress>"
```

Nützliche Befehle

Im Folgenden finden Sie eine Reihe von Befehlen, die im Rahmen der Installation von Nutzen sind.

Den Rückgabewert der Installation können Sie mit einem Skript wie dem folgenden überprüfen. Der Rückgabecode befindet sich in der Umgebungsvariablen `%errorlevel%`. Der Rückgabecode `0` bedeutet, dass der Vorgang erfolgreich war.

```
start /wait msiexec /i RaptorXMLServer.msi /q
echo %errorlevel%
```

Damit die Installation im Hintergrund, mit einem Rückgabecode und einem Log des Installationsvorgangs durchgeführt wird, führen Sie den folgenden Befehl aus:

```
start /wait msiexec /i RaptorXMLServer.msi /q /L*v! <pathToInstallLogFile>
```

Um die Installation zu ändern, führen Sie den folgenden Befehl aus:

```
msiexec /m RaptorXMLServer.msi
```

Um die Installation zu reparieren, führen Sie den folgenden Befehl aus:

```
msiexec /r RaptorXMLServer.msi
```

So deinstallieren Sie RaptorXML Server:

```
msiexec /x RaptorXMLServer.msi
```

Um RaptorXML Server im Hintergrund zu deinstallieren und das Resultat in ein detailliertes Log zu schreiben, verwenden Sie folgenden Befehl:

```
start /wait msiexec /x RaptorXMLServer.msi /q /L*v! <pathToUninstallLogFile>
```

Um RaptorXML Server in einer anderen Sprache zu installieren (die verfügbaren Sprachcodes sind: Deutsch=`de`; Spanisch=`es`; Französisch=`fr`), verwenden Sie:

```
msiexec /i RaptorXMLServer.msi INSTALLER_LANGUAGE=<languageCode>
```

Anmerkung: Auf Windows Server Core stehen die Diagramm-Funktionen von RaptorXML Server nicht zur Verfügung.

3.1.2.1 Webserver-Eigenschaften

Sie können den RaptorXML Server Web Server mit Hilfe der unten aufgelisteten Eigenschaften konfigurieren. Um eine Eigenschaft zu definieren, führen Sie den Installationsbefehl aus und hängen Sie die Eigenschaftseinstellung folgendermaßen an:

```
msiexec /i RaptorXMLServer.msi RXML_WebServer_Host=127.0.0.1
```

Liste der Eigenschaften

Eigenschaften des RaptorXML Server Web Servers:

`RXML_WebServer_Host=<IP4 Address>`
Verwenden Sie `127.0.0.1`, wenn Sie nur von diesem Rechner aus auf den Web Server zugreifen möchten. Verwenden Sie `0.0.0.0`, um globalen Zugriff auf den Web Server zu ermöglichen.

`RXML_WebServer_Port=<Port-Nummer>`
Definiert den Port für den Zugriff auf den Web Server.

`RXML_WebServer_Enabled=<0 oder 1>`
Wählen Sie `1`, damit der Rechner am aktuell definierten Port empfangsbereit ist. Wählen Sie `0`, um die Empfangsbereitschaft an diesem Port zu deaktivieren.

3.1.2.2 SSL-Webserver-Eigenschaften

Sie können den RaptorXML Server SSL-Web Server mit Hilfe der unten aufgelisteten Eigenschaften konfigurieren. Um eine Eigenschaft zu definieren, führen Sie den Installationsbefehl aus und hängen Sie die Eigenschaftseinstellung folgendermaßen an:

```
msiexec /i RaptorXMLServer.msi RXML_SSLWebServer_Host=127.0.0.1
```

Liste der Eigenschaften

Um den RaptorXML Server SSL-Web Server zu konfigurieren, verwenden Sie die folgenden Eigenschaften:

RXML_SSLWebServer_Host=<IP4 Address>
Verwenden Sie `127.0.0.1`, wenn Sie (für die verschlüsselte Übertragung) nur von diesem Rechner aus auf den SSL Web Server zugreifen möchten. Verwenden Sie `0.0.0.0`, um globalen Zugriff auf den SSL-Web Server zu ermöglichen.

RXML_SSLWebServer_Port=<Port-Nummer>
Definiert den Port für den Zugriff auf den SSL Web Server (für die verschlüsselte Übertragung).

RXML_SSLWebServer_Enabled=<0 oder 1>
Wählen Sie `1`, damit der Rechner am aktuell definierten Port empfangsbereit ist. Wählen Sie `0`, um die Empfangsbereitschaft an diesem Port zu deaktivieren.

RXML_SSLWebServer_Certificate=<Pfad-zur-Zertifikatdatei>
Der vollständige Pfad zu einem in doppelte Anführungszeichen gesetzten SSL-Zertifikat.

RXML_SSLWebServer_PrivateKey=<Pfad-zur-Private-Key-Datei>
Der vollständige Pfad zu einer in doppelte Anführungszeichen gesetzten Private Key-Datei.

3.1.2.3 Diensteigenschaften

Sie können den RaptorXML Server-Dienst mit Hilfe der unten aufgelisteten Eigenschaften konfigurieren. Um eine Eigenschaft zu definieren, führen Sie den Installationsbefehl aus und hängen Sie die Eigenschaftseinstellung folgendermaßen an:

```
msiexec /i RaptorXMLServer.msi RXML_Service_DisplayName=RaptorXMLServer
```

Liste der Eigenschaften

Um RaptorXML Server-Dienste zu konfigurieren, verwenden Sie die folgenden Eigenschaften:

RXML_Service_DisplayName=<Angezeigter Name des Diensts>
Der für den Dienst angezeigte Name. Setzen Sie den Namen in doppelte Anführungszeichen.

RXML_Service_StartType=<Starttyp>
Definiert, wie der Dienst während eines Systemstarts gestartet werden soll. Es kann einer der folgenden Werte verwendet werden: **auto | auto-delayed | demand | disabled**.

RXML_Service_Username=<Benutzername>
Definiert den angemeldeten Benutzer für den Dienst. Verwenden Sie einen der folgenden Werte: **LocalSystem | NT Authority\LocalService | NT Authority\NetworkService | <beliebiger Benutzer mit entsprechenden Rechten>**.

RXML_Service_Password=<Passwort>

Das Passwort des Benutzers, der den Dienst startet in reinem Text. (Tipp: Verwenden Sie die Benutzeroberfläche des Installationsprogramms, um Passwörter nicht als reinen Text eingeben zu müssen.) Wenn der Benutzername einer der folgenden ist, ist kein Passwort erforderlich: **`LocalSystem | NT Authority\LocalService | NT Authority\NetworkService`**.

3.1.3 Installation von LicenseServer (Windows)

Damit RaptorXML Server ausgeführt werden kann, muss das Produkt über einen Altova LicenseServer in Ihrem Netzwerk lizenziert sein. Wenn Sie RaptorXML Server oder FlowForce Server auf Windows-Systemen installieren, können Sie LicenseServer zusammen mit RaptorXML Server oder FlowForce Server installieren. Wenn in Ihrem Netzwerk bereits ein LicenseServer installiert ist, muss kein weiterer installiert werden, es sei denn, Sie benötigen eine neuere Version von LicenseServer. (*Siehe nächster Punkt, LicenseServer-Versionen.*)

Aktivieren Sie während der Installation von RaptorXML Server oder FlowForce Server je nach Bedarf die Option zum Installieren von LicenseServer.

Beachten Sie die folgenden Punkte:

- Wenn Sie Altova LicenseServer noch nicht installiert haben, belassen Sie die Standardeinstellungen unverändert. Der Assistent installiert daraufhin auf dem Rechner, auf dem Sie den Assistenten gestartet haben, die neueste Version.
- Wenn Sie LicenseServer noch nicht installiert haben und Altova LicenseServer auf einem anderen Rechner installieren möchten und diesen verwenden möchten, deaktivieren Sie das Kontrollkästchen *Altova LicenseServer auf diesem Rechner installieren* und wählen Sie **Später registrieren**. In diesem Fall müssen Sie LicenseServer separat auf dem anderen Rechner installieren und RaptorXML Server danach auf dem LicenseServer auf diesem Rechner registrieren.
- Wenn LicenseServer auf Ihrem Rechner bereits installiert wurde, aber eine niedrigere Versionsnummer als die vom Installationsassistenten angegebene hat, belassen Sie die Standardeinstellungen (zur Installation eines Upgrade auf die neuere Version) unverändert. In diesem Fall aktualisiert der Installationsassistent Ihre LicenseServer Version automatisch. Die vorhandenen Registrierungs- und Lizenzierungsdaten werden auf die neue Version von LicenseServer übertragen.
- Wenn LicenseServer bereits auf Ihrem Rechner oder in Ihrem Netzwerk installiert wurde und dieselbe Versionsnummer wie diejenige im Assistenten hat, gehen Sie folgendermaßen vor:
  - Deaktivieren Sie das Kontrollkästchen*LicenseServer auf diesem Rechner installieren.*
  - Wählen Sie unter *Dieses Produkt registrieren auf* den LicenseServer, auf dem Sie RaptorXML Server registrieren möchten. Wählen Sie alternativ dazu **Später registrieren**. Beachten Sie, dass Sie immer die Möglichkeit haben, **Später registrieren** auszuwählen, wenn Sie die LicenseServer-Verknüpfungen ignorieren möchten und mit der Installation von RaptorXML Server fortfahren möchten.

Lesen Sie nach, wie Sie RaptorXML Server auf dem Altova LicenseServer registrieren[30] und lizenzieren[31]. Nähere Informationen dazu finden Sie außerdem in der Dokumentation zu LicenseServer.

LicenseServer-Versionen

- Altova-Produkte müssen entweder (i) mit einer Version von LicenseServer, die der installierten Version von RaptorXML Server entspricht oder (ii) mit einer höheren Version von LicenseServer lizenziert werden.
- Die LicenseServer-Version, die der aktuellen Version von RaptorXML Server entspricht, ist ***3.17***.
- Unter Windows Sie können die dazugehörige Version von LicenseServer zusammen mit RaptorXML

Server installieren oder Sie können LicenseServer separat installieren. Auf Linux- und macOS-Systemen muss LicenseServer separat installiert werden.
- Bevor Sie eine neuere Version von LicenseServer installieren, muss eine eventuell vorhandene ältere Version deinstalliert werden.
- Bei der Deinstallation werden alle Registrierungs- und Lizenzierungsinformationen aus der älteren LicenseServer-Version in einer Datenbank auf Ihrem Server gespeichert. Diese Daten werden bei der Installation der neueren Version automatisch in die neuere Version importiert.
- LicenseServer-Versionen sind rückwärts kompatibel. Sie funktionieren auch mit älteren Versionen von RaptorXML Server.
- Die neueste Version von LicenseServer steht auf der Altova Website zur Verfügung. Diese Version funktioniert mit allen aktuellen oder älteren Versionen von RaptorXML Server.
- Sie finden die Versionsnummer des aktuell installierten LicenseServer am unteren Rand der LicenseServer Konfigurationsseite (alle Register).

3.1.4 Netzwerk- und Dienstkonfiguration (Windows)

Während der Installation von RaptorXML Server können Sie Einstellungen zum Aufrufen von RaptorXML Server über das Netzwerk und zur Ausführung von RaptorXML Server als Windows-Dienst konfigurieren.
Es stehen die folgenden Einstellungen zur Verfügung. Behalten Sie die Standardeinstellungen unverändert bei, falls Sie für Sie in Ordnung sind oder sich nicht sicher darüber sind. Wenn Sie eine Einstellung ändern möchten, aktivieren Sie für diese Einstellung die Schaltfläche **Ändern** (*siehe Abbildung oben*).

- Der Port für die nicht verschlüsselte Kommunikation mit RaptorXML Server.
- Ob sichere (SSL-verschlüsselte) Verbindungen mit RaptorXML Server zulässig sind. Falls ja, an welchem Port. Standardmäßig sind sichere Verbindungen deaktiviert. Nähere Informationen finden Sie im Abschnitt zum Einrichten der SSL-Verschlüsselung 279.
- Windows-Dienst-Einstellungen Dazu gehören:
  - Wie RaptorXML Server als Windows-Dienst gestartet werden soll: Automatisch, Auf Wunsch, automatisch, Automatisch verzögert oder Deaktiviert.
  - Das Benutzerkonto, das von RaptorXML Server für den Windows-Dienst verwendet werden soll: *Lokales System, Lokaler Dienst, Netzwerkdienst* oder *Anderer Benutzer*. Bei Auswahl von *Anderer Benutzer*, können Sie ähnlich wie in der Windows-Dienstverwaltungskonsole den Benutzernamen und das Passwort dieses Benutzers definieren. Beachten Sie, dass der ausgewählte Benutzer Lese/Schreibzugriff auf `C:\ProgramData\Altova` haben muss. Andernfalls könnte die Installation oder der Programmstart fehlschlagen.

Sie können die Einstellungen nach der Installation ändern. Um die Windows-Dienst-Konfiguration zu ändern, öffnen Sie die Windows-Dienstverwaltungskonsole (durch Eingabe von `Services.msc` in ein Befehlszeilenfenster) und ändern Sie den erforderlichen Dienst dort.

3.1.5 Starten von LicenseServer, RaptorXML Server (Windows)

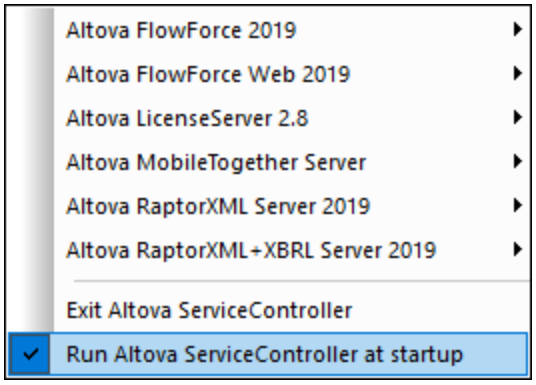
Altova LicenseServer (kurz LicenseServer) und RaptorXML Server werden beide über Altova ServiceController gestartet.

Altova ServiceController

Der Altova ServiceController (in der Folge ServiceController genannt) ist eine Applikation, mit der Sie Altova-

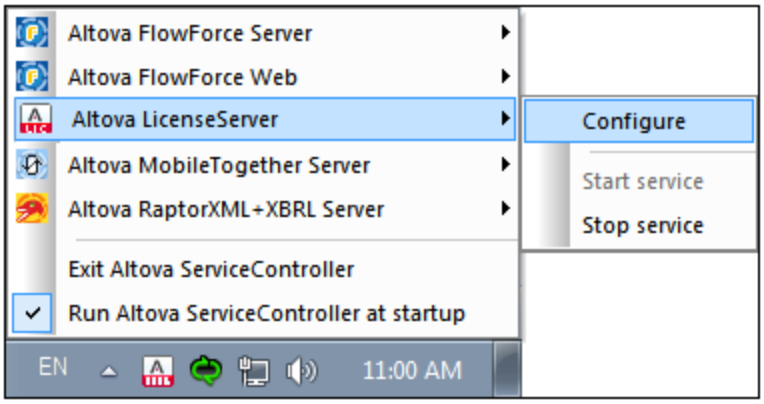
Dienste **auf Windows-Systemen** starten, beenden und konfigurieren können. ServiceController wird mit Altova LicenseServer und als Dienst installierten Altova Server-Produkten installiert (DiffDog Server, FlowForce Server, Mobile Together Server, and RaptorXML(+XBRL) Server) und kann über die Task-Leiste (*siehe Abbildung unten*) aufgerufen werden.

Damit der ServiceController automatisch ausgeführt wird, nachdem sich der Benutzer im System angemeldet hat, klicken Sie in der Task-Leiste auf das **ServiceController**-Symbol, um das **ServiceController**-Menü (*Abbildung unten*) aufzurufen und aktivieren Sie anschließend den Befehl **Run Altova ServiceController at Startup**. (Dieser Befehl ist standardmäßig aktiv). Um den ServiceController zu beenden, klicken Sie in der Task-Leiste auf das **ServiceController**-Symbol und klicken Sie im Menü, das daraufhin angezeigt wird (*Abbildung unten*) auf **Exit Altova ServiceController**.



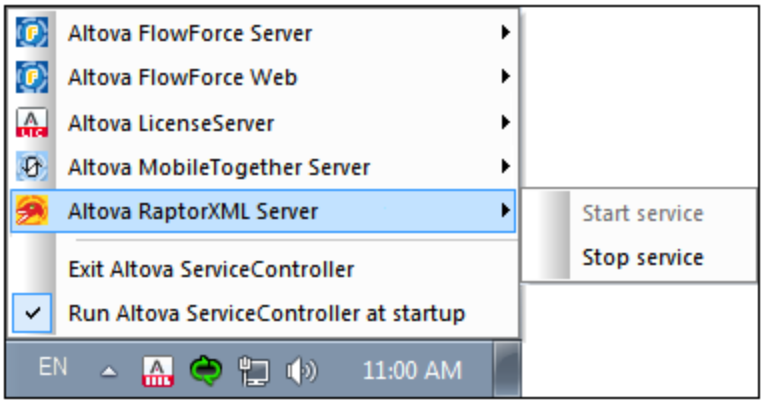
Starten von LicenseServer

Um LicenseServer zu starten, klicken Sie in der Task-Leiste auf das **ServiceController**-Symbol, positionieren Sie den Mauszeiger im angezeigten Menü über **Altova LicenseServer** (*siehe Abbildung unten*) und wählen Sie anschließend im LicenseServer Untermenü den Befehl **Start Service**. Wenn LicenseServer bereits ausgeführt wird, ist die Option *Start Service* deaktiviert. Sie können den Dienst über Service Controller auch beenden.



Starten von RaptorXML Server

Um RaptorXML Server zu starten, klicken Sie in der Task-Leiste auf das **ServiceController**-Symbol, positionieren Sie den Mauszeiger im angezeigten Menü über **Altova RaptorXML Server** (*siehe Abbildung unten*) und wählen Sie anschließend im RaptorXML Server-Untermenü den Befehl **Start Service**. Wenn RaptorXML Server bereits ausgeführt wird, ist die Option *Start Service* deaktiviert. Sie können den Dienst über Service Controller auch beenden.



Anmerkung: Wenn RaptorXML Server nur für die Ausführung von Single-Thread-Ausführungen lizenziert wurde (normalerweise, weil Ihr Rechner zwar ein Mehrkernprozessor ist, aber Ihre Lizenz nur für einen Kern gilt), dann können Sie immer nur eine Instanz von RaptorXML Server gleichzeitig verwenden: entweder als Dienst oder über die Befehlszeile. Der Grund dafür ist, dass die Einkern-Lizenz automatisch der ersten gestarteten Instanz, die gerade ausgeführt wird, zugewiesen wird; eine zweite Instanz kann erst gestartet werden, sobald die erste Instanz beendet wird.

- Wenn Sie RaptorXML Server über die Befehlszeile verwenden möchten, aber der Dienst bereits ausgeführt wird, müssen Sie den Dienst beenden, bevor Sie die Befehlszeile verwenden können.
- Wenn Sie RaptorXML Server als Dienst starten möchten, stellen Sie sicher, dass gerade keine Befehlszeilenaktion ausgeführt wird. Andernfalls können Sie den Dienst nicht starten.

3.1.6 Registrieren von RaptorXML Server (Windows)

Damit RaptorXML Server über Altova LicenseServer lizenziert werden kann, muss RaptorXML Server über einen LicenseServer in Ihrem Netzwerk lizenziert sein. Um RaptorXML Server über die Befehlszeilenschnittstelle zu registrieren, verwenden Sie den Befehl `licenseserver` und geben Sie die Adresse des LicenseServer-Rechners an (*siehe unten*).

```
RaptorXML licenseserver [options] ServerName-Or-IP-Address
```

Wenn z.B. `localhost` der Name des Servers ist, auf dem LicenseServer installiert ist, verwenden Sie den folgenden Befehl:

```
RaptorXML licenseserver localhost
```

Wenn RaptorXML Server im Rahmen von FlowForce Server installiert wurde, wird auch RaptorXML Server bei der Registrierung von FlowForce Server auf LicenseServer registriert. (i) Starten Sie Altova FlowForce Web als

Dienst über den ServiceController (*siehe vorhergehender Punkt*); (ii) Geben Sie Ihr Passwort ein, um die Setup-Seite aufzurufen; (iii) Wählen Sie den Namen oder die Adresse des LicenseServers aus und klicken Sie auf **Auf LicenseServer registrieren**. Nähere Informationen dazu finden Sie unter Registrieren von FlowForce Server.

Nachdem Sie RaptorXML Server erfolgreich registriert haben, gehen Sie zum Register "Client Management" der LicenseServer-Konfigurationsseite und weisen Sie dem Produkt eine Lizenz zu.

Nähere Informationen zum Registrieren von Altova-Produkten auf LicenseServer finden Sie im LicenseServer-Benutzerhandbuch.

3.1.7 Zuweisen einer Lizenz (Windows)

Nachdem Sie RaptorXML Server erfolgreich registriert haben, wird das Produkt auf dem Register "Client Management" der LicenseServer-Konfigurationsseite aufgelistet. Gehen Sie zu diesem Register und weisen Sie RaptorXML Server eine Lizenz zu.

Die Lizenzierung von Altova Server-Produkten basiert auf der Anzahl der auf dem Produktrechner verfügbaren Prozessorkerne. So hat z.B. ein Dual-Core-Prozessor zwei Prozessorkerne, ein Quad-Core-Prozessor vier Kerne, ein Hexa-Core-Prozessor sechs Kerne, usw. Die Anzahl der für ein Produkt lizenzierten Kerne muss größer oder gleich der Anzahl der auf diesem Serverrechner verfügbaren Kerne sein, unabhängig davon, ob es sich um einen physischen Rechner oder eine Virtual Machine handelt. Wenn ein Server z.B. acht Kerne hat (ein Octa-Core-Prozessor), müssen Sie mindestens eine Lizenz für acht Kerne erwerben. Sie können Lizenzen auch kombinieren, um das Produkt für die entsprechende Anzahl von Kernen zu lizenzieren. So können z.B. anstelle einer Lizenz für 8 Kerne auch zwei Lizenzen für 4 Kerne für einen 8-Kern-Prozessor verwendet werden.

Wenn Sie einen Server-Rechner mit einer großen Anzahl von CPU-Kernen verwenden, aber nur geringe Datenmengen verarbeiten müssen, können Sie auch eine Virtual Machine erstellen, der eine geringere Anzahl an Kernen zugewiesen ist und eine Lizenz für diese Anzahl an Kernen erwerben. In diesem Fall ist die Verarbeitungsgeschwindigkeit natürlich geringer als bei Verwendung aller Kerne des Rechners.

Anmerkung: Jede Altova Server-Produktlizenz kann immer nur für einen Client-Rechner gleichzeitig verwendet werden, selbst wenn die Lizenzkapazität dieser Lizenz noch nicht ausgeschöpft ist. (Ein Client-Rechner ist der Rechner, auf dem das Altova Server-Produkt installiert ist.) Wenn z.B. eine 10-Kern-Lizenz für einen Client-Rechner mit 6 CPU-Kernen verwendet wird, so können die verbleibenden Lizenzen für die restlichen 4 Kerne nicht gleichzeitig für einen anderen Client-Rechner verwendet werden.

Single-Thread-Ausführung
Wenn bei einem Altova-Server-Produkt eine Single-Thread-Ausführung möglich ist, so steht eine Option für die *Single-Thread-Ausführung* zur Verfügung. Wenn in solchen Fällen im Lizenzpool eine Altova Serverproduktlizenz für nur einen Prozessorkern verfügbar ist, können Sie einem Rechner mit mehreren Kernen diese Lizenz für einen Kern zuweisen. In diesem Fall führt der Rechner das Produkt an einem einzigen Kern aus. Dadurch verlangsamt sich die Verarbeitungsgeschwindigkeit, da kein Multi-Threading (welches bei mehreren Prozessorkernen möglich wäre) zur Verfügung steht. Das Produkt wird auf diesem Rechner im Single Thread-Modus ausgeführt.

Um einem Mehrkernrechner eine Lizenz für nur einen Kern zuzuweisen, aktivieren Sie in LicenseServer für das entsprechende Produkt das Kontrollkästchen *Limit to single thread execution*.

Schätzung der benötigten Prozessorkerne
Es gibt eine Reihe von externen Faktoren, die das Verarbeitungsvolumen und die Verarbeitungszeiten Ihres

Servers beeinflussen (z.B. Hardware, CPU-Auslastung, Arbeitsspeicher für andere auf dem Server laufende Applikationen). Um die Leistung möglichst genau messen zu können, empfiehlt es sich, die Applikationen in Ihrer Umgebung mit möglichst realistischen Datenvolumina und unter möglichst realistischen Bedingungen zu testen.

3.2 Einrichten unter Linux

In diesem Abschnitt werden die Installation[34] und Lizenzierung von RaptorXML Server auf Linux-Systemen (Debian, Ubuntu, CentOS, RedHat) beschrieben. Bei der Einrichtung müssen die folgenden Schritte durchgeführt werden:

1. Installation von RaptorXML Server[34]
2. Installation von LicenseServer[35]
3. Starten von LicenseServer[36]
4. Registrieren von RaptorXML Server auf LicenseServer[37]
5. Zuweisen einer Lizenz zu RaptorXML Server[37]

Die einzelnen oben beschriebenen Schritte müssen nicht in genau derselben Reihenfolge wie aufgelistet erfolgen. Bevor Sie beginnen, muss jedoch zunächst die Installation durchgeführt werden. Außerdem muss RaptorXML Server auf LicenseServer registriert werden, bevor Sie RaptorXML Server über LicenseServer eine Lizenz zuweisen können.

Systemvoraussetzungen (Linux)

- Red Hat Enterprise Linux 7 oder neuer
- CentOS 7, CentOS Stream 8
- Debian 10 oder neuer
- Ubuntu 20.04, 22.04, 24.04
- AlmaLinux 9.0
- Rocky Linux 9.0

Voraussetzungen

- Führen Sie die Installation entweder als **root**-Benutzer durch oder als Benutzer mit **sudo**-Rechten.
- Die vorherige Version von RaptorXML Server muss deinstalliert werden, bevor Sie eine neue Version installieren.
- Wenn Sie beabsichtigen, die Diagrammfunktionalität von Altova zu verwenden, muss auf Ihrem System mindestens eine Schriftart installiert sein, damit die Diagramme korrekt dargestellt werden können. Installierte Schriftarten können Sie z.B. mit dem Befehl **`fc-list`** aus der Fontconfig-Bibliothek auflisten.
- Um die Applikation installieren und ausführen zu können, werden die folgenden Bibliotheken benötigt. Falls die unten angeführten Pakete auf Ihrem Linux-Rechner noch nicht zur Verfügung stehen, führen Sie die Befehl `yum` (oder ggf. `apt-get`) aus, um sie zu installieren.

CentOS, RedHat	Debian	Ubuntu
krb5-libs	libgssapi-krb5-2	libgssapi-krb5-2

3.2.1 Installation unter Linux

RaptorXML Server steht für die Installation auf Linux-Systemen zur Verfügung. Führen Sie die Installation entweder als `root`-Benutzer durch oder als Benutzer mit `sudo`-Rechten.

Integration von FlowForce Server mit anderen Altova Server-Produkten

Wenn Sie RaptorXML Server zusammen mit FlowForce Server installieren, sollten Sie zuerst FlowForce Server installieren. Wenn Sie RaptorXML Server vor FlowForce Server installiert haben, führen Sie im Anschluss an die Installation von RaptorXML Server und FlowForce Server den folgenden Befehl aus:

```
cp /opt/Altova/RaptorXMLServer2025/etc/*.tool /opt/Altova/FlowForceServer2025/tools
```

Dieser Befehl kopiert die **`.tool`**-Datei aus dem Verzeichnis **`/etc`** von RaptorXML Server in das FlowForce Server **`/tools`**-Verzeichnis. Die Datei **`.tool`** wird von FlowForce Server benötigt. Sie enthält den Pfad zur ausführbaren RaptorXML Server-Datei. Sie müssen diesen Befehl nicht ausführen, wenn Sie FlowForce Server vor RaptorXML Server installieren.

Deinstallieren von RaptorXML Server

Bevor Sie RaptorXML Server installieren, sollten Sie ältere Versionen deinstallieren.

So überprüfen Sie, welche Altova Server-Produkte aktuell installiert sind:

```
[Debian, Ubuntu]:   dpkg --list | grep Altova
[CentOS, RedHat]:   rpm -qa | grep server
```

So deinstallieren Sie eine alte Version von RaptorXML Server:

```
[Debian, Ubuntu]:   sudo dpkg --remove raptorxmlserver
[CentOS, RedHat]:   sudo rpm -e raptorxmlserver
```

Auf Debian- und Ubuntu-Systemen kann es vorkommen, dass RaptorXML Server auch nach seiner Deinstallation noch in der Liste der installierten Produkte angezeigt wird. Führen Sie in diesem Fall den **`purge`**-Befehl aus, um RaptorXML Server aus der Liste zu entfernen. Anstelle des oben aufgelisteten Befehls **`remove`** *können Sie auch den* **`purge`**-Befehl verwenden.

```
[Debian, Ubuntu]:   sudo dpkg --purge raptorxmlserver
```

Herunterladen des RaptorXML Server Linux-Pakets

Auf der Altova Website stehen RaptorXML Server-Installationspakete für die folgenden Linux-Systeme zur Verfügung.

Distribution	Paketerweiterung
Debian	`.deb`
Ubuntu	`.deb`
CentOS	`.rpm`

Distribution	Paketerweiterung
RedHat	`.rpm`

Nachdem Sie das Linux-Paket heruntergeladen haben, kopieren Sie das Paket in ein beliebiges Verzeichnis auf dem Linux-System. Da RaptorXML Server auf einen Altova LicenseServer lizenziert werden muss, benötigen, sollten Sie gleichzeitig mit RaptorXML Server auch LicenseServer von der Altova Website herunterladen.

Installieren von RaptorXML Server

Wechseln Sie in einem Terminal-Fenster zu dem Verzeichnis, in das Sie das Linux-Paket kopiert haben. Wenn Sie es z.B in ein Benutzerverzeichnis namens `MyAltova` (z.B. im Verzeichnis `/home/User`) kopiert haben, dann wechseln Sie folgendermaßen zu diesem Verzeichnis:

```
cd /home/User/MyAltova
```

Installieren Sie RaptorXML Server mit dem entsprechenden Befehl:

```
[Debian]:   sudo dpkg --install raptorxml-2025-debian.deb
[Ubuntu]:   sudo dpkg --install raptorxml-2025-ubuntu.deb
[CentOS]:   sudo rpm -ivh raptorxml-2025-1.x86_64.rpm
[RedHat]:   sudo rpm -ivh raptorxml-2025-1.x86_64.rpm
```

Sie müssen den Namen des obigen Pakets eventuell anpassen, damit er der aktuellen Release- oder Service Pack-Version entspricht.

Das RaptorXML Server-Paket wird im folgenden Ordner installiert:

```
/opt/Altova/RaptorXMLServer2025
```

3.2.2 Installation von LicenseServer (Linux)

Damit RaptorXML Server ausgeführt werden kann, muss das Produkt über einen Altova LicenseServer in Ihrem Netzwerk lizenziert sein. Laden Sie Altova LicenseServer von der Altova Website herunter und kopieren Sie das Paket in ein beliebiges Verzeichnis. Installieren Sie es genau wie RaptorXML Server (*siehe vorheriges Kapitel* 34).

```
[Debian]:   sudo dpkg --install licenseserver-3.17-debian.deb
[Ubuntu]:   sudo dpkg --install licenseserver-3.17-ubuntu.deb
[CentOS]:   sudo rpm -ivh licenseserver-3.17-1.x86_64.rpm
[RedHat]:   sudo rpm -ivh licenseserver-3.17-1.x86_64.rpm
```

Das LicenseServer-Paket wird im folgenden Ordner installiert:

```
/opt/Altova/LicenseServer
```

Lesen Sie nach, wie Sie RaptorXML Server auf dem Altova LicenseServer registrieren 37 und lizenzieren 37. Nähere Informationen dazu finden Sie außerdem in der Dokumentation zu LicenseServer.

LicenseServer-Versionen

- Altova-Produkte müssen entweder (i) mit einer Version von LicenseServer, die der installierten Version von RaptorXML Server entspricht oder (ii) mit einer höheren Version von LicenseServer lizenziert werden.
- Die LicenseServer-Version, die der aktuellen Version von RaptorXML Server entspricht, ist ***3.17***.
- Unter Windows Sie können die dazugehörige Version von LicenseServer zusammen mit RaptorXML Server installieren oder Sie können LicenseServer separat installieren. Auf Linux- und macOS-Systemen muss LicenseServer separat installiert werden.
- Bevor Sie eine neuere Version von LicenseServer installieren, muss eine eventuell vorhandene ältere Version deinstalliert werden.
- Bei der Deinstallation werden alle Registrierungs- und Lizenzierungsinformationen aus der älteren LicenseServer-Version in einer Datenbank auf Ihrem Server gespeichert. Diese Daten werden bei der Installation der neueren Version automatisch in die neuere Version importiert.
- LicenseServer-Versionen sind rückwärts kompatibel. Sie funktionieren auch mit älteren Versionen von RaptorXML Server.
- Die neueste Version von LicenseServer steht auf der Altova Website zur Verfügung. Diese Version funktioniert mit allen aktuellen oder älteren Versionen von RaptorXML Server.
- Sie finden die Versionsnummer des aktuell installierten LicenseServer am unteren Rand der LicenseServer Konfigurationsseite (alle Register).

3.2.3 Starten von LicenseServer, RaptorXML Server (LInux)

Starten Sie LicenseServer und RaptorXML Server entweder als `root`-Benutzer oder als Benutzer mit `sudo`-Rechten.

Starten von LicenseServer

Um RaptorXML Server korrekt auf LicenseServer zu registrieren und zu lizenzieren, muss LicenseServer als Daemon im Netzwerk ausgeführt werden. Starten Sie LicenseServer mit dem folgenden Befehl als Daemon:

```
sudo systemctl start licenseserver
```

(Um LicenseServer zu beenden, ersetzen Sie im obigen Befehl **`start`** durch **`stop`**.) Beispiel:

```
sudo systemctl stop licenseserver
```

Starten von RaptorXML Server

Starten Sie RaptorXML Server mit dem folgenden Befehl als Daemon:

```
sudo systemctl start raptorxmlserver
```

Um RaptorXML Server zu beenden, ersetzen Sie im obigen Befehl **`start`** durch **`stop`**. Beispiel:

```
sudo systemctl stop raptorxmlserver
```

Überprüfen des Status von Daemonen

Um zu überprüfen, ob ein Daemon ausgeführt wird, führen Sie den folgenden Befehl aus und ersetzen Sie **<servicename>** durch den Namen des zu überprüfenden Daemons:

```
sudo service <servicename> status
```

3.2.4 Registrieren von RaptorXML Server (Linux)

Damit RaptorXML Server über Altova LicenseServer lizenziert werden kann, muss RaptorXML Server über einen LicenseServer in Ihrem Netzwerk lizenziert sein.

Um RaptorXML Server zu registrieren, rufen Sie seine CLI auf und verwenden Sie den Befehl `licenseserver`:

```
sudo /opt/Altova/RaptorXMLServer2025/bin/raptorxml licenseserver [options] ServerName-Or-IP-Address
```

Wenn z.B. `localhost` der Name des Servers ist, auf dem LicenseServer installiert ist:

```
sudo /opt/Altova/RaptorXMLServer2025/bin/raptorxml licenseserver localhost
```

Im obigen Befehl ist `localhost` der Name des Servers, auf dem LicenseServer installiert ist. Beachten Sie, dass der Pfad der ausführbaren RaptorXML Server-Datei der folgende ist:

```
/opt/Altova/RaptorXMLServer2025/bin/
```

Nachdem Sie RaptorXML Server erfolgreich registriert haben, gehen Sie zum Register "Client Management" der LicenseServer-Konfigurationsseite und weisen Sie dem Produkt eine Lizenz zu.

Nähere Informationen zum Registrieren von Altova-Produkten auf LicenseServer finden Sie im LicenseServer-Benutzerhandbuch.

3.2.5 Zuweisen einer Lizenz (LInux)

Nachdem Sie RaptorXML Server erfolgreich registriert haben, wird das Produkt auf dem Register "Client Management" der LicenseServer-Konfigurationsseite aufgelistet. Gehen Sie zu diesem Register und weisen Sie RaptorXML Server eine Lizenz zu.

Die Lizenzierung von Altova Server-Produkten basiert auf der Anzahl der auf dem Produktrechner verfügbaren Prozessorkerne. So hat z.B. ein Dual-Core-Prozessor zwei Prozessorkerne, ein Quad-Core-Prozessor vier Kerne, ein Hexa-Core-Prozessor sechs Kerne, usw. Die Anzahl der für ein Produkt lizenzierten Kerne muss größer oder gleich der Anzahl der auf diesem Serverrechner verfügbaren Kerne sein, unabhängig davon, ob es sich um einen physischen Rechner oder eine Virtual Machine handelt. Wenn ein Server z.B. acht Kerne hat (ein Octa-Core-Prozessor), müssen Sie mindestens eine Lizenz für acht Kerne erwerben. Sie können Lizenzen auch kombinieren, um das Produkt für die entsprechende Anzahl von Kernen zu lizenzieren. So können z.B. anstelle einer Lizenz für 8 Kerne auch zwei Lizenzen für 4 Kerne für einen 8-Kern-Prozessor verwendet werden.

Wenn Sie einen Server-Rechner mit einer großen Anzahl von CPU-Kernen verwenden, aber nur geringe Datenmengen verarbeiten müssen, können Sie auch eine Virtual Machine erstellen, der eine geringere Anzahl an Kernen zugewiesen ist und eine Lizenz für diese Anzahl an Kernen erwerben. In diesem Fall ist die Verarbeitungsgeschwindigkeit natürlich geringer als bei Verwendung aller Kerne des Rechners.

Anmerkung: Jede Altova Server-Produktlizenz kann immer nur für einen Client-Rechner gleichzeitig verwendet werden, selbst wenn die Lizenzkapazität dieser Lizenz noch nicht ausgeschöpft ist. (Ein Client-Rechner ist der Rechner, auf dem das Altova Server-Produkt installiert ist.) Wenn z.B. eine 10-Kern-Lizenz für einen Client-Rechner mit 6 CPU-Kernen verwendet wird, so können die verbleibenden Lizenzen für die restlichen 4 Kerne nicht gleichzeitig für einen anderen Client-Rechner verwendet werden.

Single-Thread-Ausführung
Wenn bei einem Altova-Server-Produkt eine Single-Thread-Ausführung möglich ist, so steht eine Option für die *Single-Thread-Ausführung* zur Verfügung. Wenn in solchen Fällen im Lizenzpool eine Altova Serverproduktlizenz für nur einen Prozessorkern verfügbar ist, können Sie einem Rechner mit mehreren Kernen diese Lizenz für einen Kern zuweisen. In diesem Fall führt der Rechner das Produkt an einem einzigen Kern aus. Dadurch verlangsamt sich die Verarbeitungsgeschwindigkeit, da kein Multi-Threading (welches bei mehreren Prozessorkernen möglich wäre) zur Verfügung steht. Das Produkt wird auf diesem Rechner im Single Thread-Modus ausgeführt.

Um einem Mehrkernrechner eine Lizenz für nur einen Kern zuzuweisen, aktivieren Sie in LicenseServer für das entsprechende Produkt das Kontrollkästchen *Limit to single thread execution*.

Schätzung der benötigten Prozessorkerne
Es gibt eine Reihe von externen Faktoren, die das Verarbeitungsvolumen und die Verarbeitungszeiten Ihres Servers beeinflussen (z.B. Hardware, CPU-Auslastung, Arbeitsspeicher für andere auf dem Server laufende Applikationen). Um die Leistung möglichst genau messen zu können, empfiehlt es sich, die Applikationen in Ihrer Umgebung mit möglichst realistischen Datenvolumina und unter möglichst realistischen Bedingungen zu testen.

3.3 Einrichten auf macOS

In diesem Abschnitt werden die Installation[39] und Lizenzierung von RaptorXML Server auf macOS-Systemen beschrieben. Bei der Einrichtung müssen die folgenden Schritte durchgeführt werden:

1. Installation von RaptorXML Server[39]
2. Installation von LicenseServer[41]
3. Starten von LicenseServer[41]
4. Registrieren von RaptorXML Server auf LicenseServer[42]
5. Zuweisen einer Lizenz zu RaptorXML Server[42]

Die einzelnen oben beschriebenen Schritte müssen nicht in genau derselben Reihenfolge wie aufgelistet erfolgen. Bevor Sie beginnen, muss jedoch zunächst die Installation durchgeführt werden. Außerdem muss RaptorXML Server auf LicenseServer registriert werden, bevor Sie RaptorXML Server über LicenseServer eine Lizenz zuweisen können.

Systemvoraussetzungen (macOS)

Beachten Sie die folgende Systemvoraussetzung:

- macOS 12 oder neuer

Voraussetzungen

Beachten Sie dabei die folgenden Voraussetzungen:

- Stellen Sie sicher, dass Altova LicenseServer installiert wurde und ausgeführt wird.
- Führen Sie die Installation entweder als `root`-Benutzer durch oder als Benutzer mit `sudo`-Rechten.
- Die vorherige Version von RaptorXML Server muss deinstalliert werden, bevor Sie eine neue Version installieren.
- Wenn Sie beabsichtigen, die Diagrammfunktionalität von Altova zu verwenden, muss auf Ihrem System mindestens eine Schriftart installiert sein, damit die Diagramme korrekt dargestellt werden können. Installierte Schriftarten können Sie z.B. mit dem Befehl **`fc-list`** aus der Fontconfig-Bibliothek auflisten.
- Der macOS-Rechner muss so konfiguriert sein, dass sein Name zu einer IP-Adresse aufgelöst wird (d.h. der Host-Name muss vom Terminal aus mit dem Befehl **`ping <hostname>`** erfolgreich angepingt werden können).

3.3.1 Installation auf macOS

In diesem Kapitel werden die Installation und Konfiguration von RaptorXML Server auf macOS-Systemen beschrieben.

Integration mit FlowForce

Wenn Sie RaptorXML Server zusammen mit FlowForce Server installieren, sollten Sie zuerst FlowForce Server installieren. Wenn Sie RaptorXML Server vor FlowForce Server installiert haben, führen Sie im Anschluss an die Installation der beiden Produkte den folgenden Befehl aus:

```
cp /usr/local/Altova/RaptorXMLServer2025/etc/*.tool /usr/local/Altova/FlowForceServer2025/tools
```

Dieser Befehl kopiert die `.tool`-Datei aus dem Verzeichnis `/etc` von RaptorXML Server in das FlowForce Server `/tools`-Verzeichnis. Die Datei `.tool` wird von FlowForce Server benötigt. Sie enthält den Pfad zur ausführbaren RaptorXML Server-Datei. Sie müssen diesen Befehl nicht ausführen, wenn Sie FlowForce Server vor RaptorXML Server installieren.

Deinstallieren von RaptorXML Server

Bevor Sie RaptorXML Server deinstallieren, beenden Sie den Dienst mit dem folgenden Befehl:

```
sudo launchctl unload /Library/LaunchDaemons/com.altova.RaptorXMLServer2025.plist
```

Um zu überprüfen, ob der Dienst beendet wurde, öffnen Sie den Activity Monitor im Finder und stellen Sie sicher, dass sich RaptorXML Server nicht in der Liste befindet. Klicken Sie im Finder unter Applications mit der rechten Maustaste auf das RaptorXML Server-Symbol und wählen Sie den Befehl **Move to Trash**. Die Applikation wird daraufhin in den Papierkorb verschoben. Sie müssen die Applikation jetzt noch aus dem Ordner `usr` entfernen. Führen Sie dazu den folgenden Befehl aus:

```
sudo rm -rf /usr/local/Altova/RaptorXMLServer2025/
```

Wenn Sie eine ältere Version von Altova LicenseServer deinstallieren müssen, beenden Sie den Dienst zuerst mit dem folgenden Befehl:

```
sudo launchctl unload /Library/LaunchDaemons/com.altova.LicenseServer.plist
```

Um zu überprüfen, ob ein Dienst beendet wurde, öffnen Sie den Activity Monitor im Finder und stellen Sie sicher, dass sich LicenseServer nicht in der Liste befindet. Fahren Sie anschließend mit der Deinstallation fort, wie oben für RaptorXML Server beschrieben.

Installieren von RaptorXML Server

Um RaptorXML Server zu installieren, gehen Sie folgendermaßen vor:

1. Laden Sie die `.dmg` (Disk Image)-Datei von RaptorXML Server von der Altova Website (https://www.altova.com/de/download.html) in ein lokales Verzeichnis herunter.
2. Klicken Sie auf die heruntergeladene Disk Image-Datei (`.dmg`), um sie zu öffnen. Dadurch wird der RaptorXML Server Installer als neues virtuelles Laufwerk auf Ihrem Computer angezeigt.
3. Doppelklicken Sie in diesem neuen virtuellen Laufwerk auf das Installer-Paket (`.pkg`).
4. Befolgen Sie die selbsterklärenden Anweisungen des Installationsassistenten, in dem Sie auch die Lizenzvereinbarung akzeptieren müssen, bevor Sie mit der Installation fortfahren können.
5. Um das Laufwerk nach der Installation auszuwerfen, klicken Sie mit der rechten Maustaste darauf und wählen Sie **Eject**.

Das RaptorXML Server-Paket wird im folgenden Ordner installiert:

`/usr/local/Altova/RaptorXMLServer2025` (Applikationsbinärdateien)
`/var/Altova/RaptorXMLServer` (Datendateien: Datenbank und Logs)

Der RaptorXML Server Server-Daemon wird nach einer Installation und einem Neustart des Rechners automatisch gestartet. Sie können RaptorXML Server jederzeit mit dem folgenden Befehl als Daemon starten:

```
sudo launchctl load /Library/LaunchDaemons/com.altova.RaptorXMLServer2025.plist
```

3.3.2 Installation von LicenseServer (macOS)

Altova LicenseServer kann von der Altova Website (https://www.altova.com/de/download.html) heruntergeladen werden. Führen Sie die Installation durch, wie hier 39 beschrieben.

Das LicenseServer-Paket wird im folgenden Ordner installiert:

```
/usr/local/Altova/LicenseServer
```

Lesen Sie nach, wie Sie RaptorXML Server auf dem Altova LicenseServer registrieren 42 und lizenzieren 42. Nähere Informationen dazu finden Sie außerdem in der Dokumentation zu LicenseServer.

LicenseServer-Versionen

- Altova-Produkte müssen entweder (i) mit einer Version von LicenseServer, die der installierten Version von RaptorXML Server entspricht oder (ii) mit einer höheren Version von LicenseServer lizenziert werden.
- Die LicenseServer-Version, die der aktuellen Version von RaptorXML Server entspricht, ist ***3.17***.
- Unter Windows Sie können die dazugehörige Version von LicenseServer zusammen mit RaptorXML Server installieren oder Sie können LicenseServer separat installieren. Auf Linux- und macOS-Systemen muss LicenseServer separat installiert werden.
- Bevor Sie eine neuere Version von LicenseServer installieren, muss eine eventuell vorhandene ältere Version deinstalliert werden.
- Bei der Deinstallation werden alle Registrierungs- und Lizenzierungsinformationen aus der älteren LicenseServer-Version in einer Datenbank auf Ihrem Server gespeichert. Diese Daten werden bei der Installation der neueren Version automatisch in die neuere Version importiert.
- LicenseServer-Versionen sind rückwärts kompatibel. Sie funktionieren auch mit älteren Versionen von RaptorXML Server.
- Die neueste Version von LicenseServer steht auf der Altova Website zur Verfügung. Diese Version funktioniert mit allen aktuellen oder älteren Versionen von RaptorXML Server.
- Sie finden die Versionsnummer des aktuell installierten LicenseServer am unteren Rand der LicenseServer Konfigurationsseite (alle Register).

3.3.3 Starten von LicenseServer, RaptorXML Server (macOS)

Starten Sie LicenseServer und RaptorXML Server entweder als `root`-Benutzer oder als Benutzer mit `sudo`-Rechten.

Starten von LicenseServer

Um RaptorXML Server korrekt auf LicenseServer zu registrieren und zu lizenzieren, muss LicenseServer als Daemon ausgeführt werden. Starten Sie LicenseServer mit dem folgenden Befehl als Daemon:

```
sudo launchctl load /Library/LaunchDaemons/com.altova.LicenseServer.plist
```

Um LicenseServer zu beenden, ersetzen Sie im obigen Befehl **load** durch **unload**.)

Starten von RaptorXML Server

Der RaptorXML Server Server-Daemon wird nach einer Installation und einem Neustart des Rechners automatisch gestartet. Sie können RaptorXML Server mit dem folgenden Befehl als Daemon starten:

```
sudo launchctl load /Library/LaunchDaemons/com.altova.RaptorXMLServer.plist
```

Um RaptorXML Server zu beenden, verwenden Sie den folgenden Befehl:

```
sudo launchctl unload /Library/LaunchDaemons/com.altova.RaptorXMLServer.plist
```

3.3.4 Registrieren von RaptorXML Server (macOS)

Damit RaptorXML Server über Altova LicenseServer lizenziert werden kann, muss RaptorXML Server über einen LicenseServer in Ihrem Netzwerk lizenziert sein.

Um RaptorXML Server über die Befehlszeile zu registrieren, verwenden Sie den Befehl `licenseserver`:

```
sudo /usr/local/Altova/RaptorXMLServer2025/bin/RaptorXML licenseserver [options]
ServerName-Or-IP-Address
```

Wenn z.B. `localhost` der Name des Servers ist, auf dem LicenseServer installiert ist:

```
sudo /usr/local/Altova/RaptorXMLServer2025/bin/RaptorXML licenseserver localhost
```

Im obigen Befehl ist `localhost` der Name des Servers, auf dem LicenseServer installiert ist. Beachten Sie, dass der Pfad der ausführbaren RaptorXML Server-Datei der folgende ist:

```
/usr/local/Altova/RaptorXMLServer2025/bin/
```

Nachdem Sie RaptorXML Server erfolgreich registriert haben, gehen Sie zum Register "Client Management" der LicenseServer-Konfigurationsseite und weisen Sie dem Produkt eine Lizenz zu.

Nähere Informationen zum Registrieren von Altova-Produkten auf LicenseServer finden Sie im LicenseServer-Benutzerhandbuch.

3.3.5 Zuweisen einer Lizenz (macOS)

Nachdem Sie RaptorXML Server erfolgreich registriert haben, wird das Produkt auf dem Register "Client Management" der LicenseServer-Konfigurationsseite aufgelistet. Gehen Sie zu diesem Register und weisen Sie RaptorXML Server eine Lizenz zu.

Die Lizenzierung von Altova Server-Produkten basiert auf der Anzahl der auf dem Produktrechner verfügbaren Prozessorkerne. So hat z.B. ein Dual-Core-Prozessor zwei Prozessorkerne, ein Quad-Core-Prozessor vier Kerne, ein Hexa-Core-Prozessor sechs Kerne, usw. Die Anzahl der für ein Produkt lizenzierten Kerne muss größer oder gleich der Anzahl der auf diesem Serverrechner verfügbaren Kerne sein, unabhängig davon, ob es

sich um einen physischen Rechner oder eine Virtual Machine handelt. Wenn ein Server z.B. acht Kerne hat (ein Octa-Core-Prozessor), müssen Sie mindestens eine Lizenz für acht Kerne erwerben. Sie können Lizenzen auch kombinieren, um das Produkt für die entsprechende Anzahl von Kernen zu lizenzieren. So können z.B. anstelle einer Lizenz für 8 Kerne auch zwei Lizenzen für 4 Kerne für einen 8-Kern-Prozessor verwendet werden.

Wenn Sie einen Server-Rechner mit einer großen Anzahl von CPU-Kernen verwenden, aber nur geringe Datenmengen verarbeiten müssen, können Sie auch eine Virtual Machine erstellen, der eine geringere Anzahl an Kernen zugewiesen ist und eine Lizenz für diese Anzahl an Kernen erwerben. In diesem Fall ist die Verarbeitungsgeschwindigkeit natürlich geringer als bei Verwendung aller Kerne des Rechners.

Anmerkung: Jede Altova Server-Produktlizenz kann immer nur für einen Client-Rechner gleichzeitig verwendet werden, selbst wenn die Lizenzkapazität dieser Lizenz noch nicht ausgeschöpft ist. (Ein Client-Rechner ist der Rechner, auf dem das Altova Server-Produkt installiert ist.) Wenn z.B. eine 10-Kern-Lizenz für einen Client-Rechner mit 6 CPU-Kernen verwendet wird, so können die verbleibenden Lizenzen für die restlichen 4 Kerne nicht gleichzeitig für einen anderen Client-Rechner verwendet werden.

Single-Thread-Ausführung
Wenn bei einem Altova-Server-Produkt eine Single-Thread-Ausführung möglich ist, so steht eine Option für die *Single-Thread-Ausführung* zur Verfügung. Wenn in solchen Fällen im Lizenzpool eine Altova Serverproduktlizenz für nur einen Prozessorkern verfügbar ist, können Sie einem Rechner mit mehreren Kernen diese Lizenz für einen Kern zuweisen. In diesem Fall führt der Rechner das Produkt an einem einzigen Kern aus. Dadurch verlangsamt sich die Verarbeitungsgeschwindigkeit, da kein Multi-Threading (welches bei mehreren Prozessorkernen möglich wäre) zur Verfügung steht. Das Produkt wird auf diesem Rechner im Single Thread-Modus ausgeführt.

Um einem Mehrkernrechner eine Lizenz für nur einen Kern zuzuweisen, aktivieren Sie in LicenseServer für das entsprechende Produkt das Kontrollkästchen *Limit to single thread execution*.

Schätzung der benötigten Prozessorkerne
Es gibt eine Reihe von externen Faktoren, die das Verarbeitungsvolumen und die Verarbeitungszeiten Ihres Servers beeinflussen (z.B. Hardware, CPU-Auslastung, Arbeitsspeicher für andere auf dem Server laufende Applikationen). Um die Leistung möglichst genau messen zu können, empfiehlt es sich, die Applikationen in Ihrer Umgebung mit möglichst realistischen Datenvolumina und unter möglichst realistischen Bedingungen zu testen.

3.4 Upgraden von RaptorXML Server

Am einfachsten lässt sich die Lizenz aus der vorherigen Version von RaptorXML Server bei der Installation auf die neuere Version übertragen: Die wichtigsten Schritte bei der Installation sind die folgenden:

1. Registrieren Sie die neue Version von RaptorXML Server auf dem LicenseServer, auf dem sich die Lizenz der älteren Version von RaptorXML Server befindet.
2. Akzeptieren Sie die Lizenzvereinbarung von RaptorXML Server. (Wenn Sie der Vereinbarung nicht zustimmen, wird die neue Version nicht installiert.)

Anmerkung: Wenn Sie RaptorXML Server nicht während der Installation auf LicenseServer registrieren, können Sie dies später nachholen und die Lizenzierung erst dann abschließen.

3.5 Migrieren von RaptorXML Server auf einen neuen Rechner

Wenn Sie RaptorXML Server von einem Rechner auf einen anderen (eventuell auch auf eine andere Plattform) migrieren möchten, befolgen Sie die Richtlinien weiter unten.

Bei der Migration von RaptorXML Server auf einen neuen Rechner wird die Lizenz vom alten Rechner einem neuen Rechner zugewiesen. Gehen Sie dazu folgendermaßen vor:

1. Installieren Sie RaptorXML Server auf dem neuen Rechner. Wenn das Produkt bereits im Rahmen der FlowForce Server-Installation installiert wurde, ignorieren Sie diesen Schritt.
2. Registrieren Sie RaptorXML Server auf dem neuen Rechner auf Altova LicenseServer.
3. Stellen Sie sicher, dass der Server auf dem alten Rechner von keinen Clients verwendet wird.
4. Öffnen Sie die Altova LicenseServer-Verwaltungsseite. Deaktivieren Sie die Lizenz des alten RaptorXML Server Rechners und weisen Sie sie dem neuen Rechner zu.

Anmerkung: Migrieren Sie die Server-Konfigurationsdatei, um Ihre vorherigen Konfigurationseinstellungen beizubehalten.

Anmerkung: Wenn Sie auf dem alten Rechner XML-Kataloge verwendet haben, migrieren Sie diese auf den neuen Rechner.

3.6 Sicherheitsaspekte

XSLT, XPath und XQuery sind Turing-vollständige funktionale Programmiersprachen mit Zugriff auf lokale und entfernte Dateien und der Möglichkeit zur dynamischen Ausführung. Es wird daher empfohlen, den Zugriff auf diese Dateien für Transformationen und/oder die Dateiverarbeitung nur in einer sicheren und geregelten Umgebung, in der die Kontrolle über die Input-Dateien gewährleistet ist und sichergestellt werden kann, dass nur zuvor überprüfte Scripts ausgeführt werden, zu gestatten. Sollte die Notwendigkeit bestehen, diese Dateien von einem externen/öffentlichen Netzwerk (oder einem nicht sicheren Sub-Netzwerk) aufzurufen, wird empfohlen, den Zugriff mit Hilfe eines Reverse Proxy Servers, der eine Benutzerauthentifizierung und -autorisierung implementiert, einzuschränken. Außerdem wird empfohlen, den Vorgang über ein separates Benutzerkonto auszuführen, bei dem der Zugriff auf Betriebssystemebene konfiguriert ist, um den Zugriff auf die autorisierten Bereiche des Dateisystems einzuschränken.

4 Allgemeine Verfahren

RaptorXML verfügt über spezielle Optionen, die XML-Kataloge[48] und globale Altova-Ressourcen[55] unterstützen. Beides davon verbessert die Portabilität und Modularität. Sie können diese Funktionalitäten daher in Ihrer eigenen Umgebung zu Ihrem Vorteil einsetzen.

Dieser Abschnitt enthält die folgenden Beschreibungen:

- Verwendung von XML-Katalogen[48].
- Arbeiten mit globalen Altova-Ressourcen[55].
- Sicherheitsfragen[57] im Zusammenhang mit RaptorXML-Verfahren und Lösungsmöglichkeiten.

4.1 XML-Kataloge

Mit Hilfe des XML-Katalogmechanismus können Dateien aus lokalen Ordnern abgerufen und so die Verarbeitungsgeschwindigkeit gesteigert werden. Gleichzeitig wird dadurch die Übertragbarkeit von Dokumenten verbessert, da in diesem Fall nur die Katalogdatei-URIs geändert werden müssen. Nähere Informationen dazu finden Sie im Abschnitt Funktionsweise von Katalogen 48.

In den XML-Produkten von Altova kommt ein Katalogmechanismus zur Anwendung, mit dem gemeinsam verwendete Dateien wie DTDS und XML-Schemas schnell aufgerufen und geladen werden können. Dieser Katalogmechanismus kann vom Benutzer angepasst und erweitert werden und ist in den Abschnitten Katalogstruktur in RaptorXML Server 50 und Anpassen von Katalogen 51 beschrieben. Im Abschnitt Variablen für Systempfade 53 sind Windows-Variablen für häufig benötigte Systempfade aufgelistet. Diese Variablen können in Katalogdateien verwendet werden, um gemeinsam verwendete Ordner aufzurufen.

Dieser Abschnitt ist in die folgenden Unterabschnitte gegliedert:

Nähere Informationen zu Katalogen finden Sie in der XML-Katalogspezifikation.

Installation von Schemas über Schema Manager

Mit Hilfe von Schema-Manager 402 können Sie wichtige Schemas schnell und einfach installieren und Konfigurations-Katalogdateien einrichten, um diese installierten Schemas aufzurufen. Nähere Informationen dazu finden Sie im Abschnitt Schema-Manager 402.

Wenn ein Dokument anhand eines nicht installierten, aber über Schema-Manager 402 verfügbaren Schemas validiert wird, wird die Installation über Schema-Manager automatisch gestartet. Wenn das über Schema-Manager zu installierende Schema-Paket jedoch Namespace-Zuordnungen enthält, wird das Schema nicht automatisch installiert; in diesem Fall müssen Sie Schema-Manager starten, das/die gewünschte(n) Paket(e) auswählen und die Installation starten. Wenn RaptorXML Server eine Schema-Komponente nicht korrekt findet, starten Sie RaptorXML Server neu und versuchen Sie es erneut.

4.1.1 Funktionsweise von Katalogen

Mit Hilfe von Katalogen können Umleitungen sowohl zu DTDs als auch XML-Schemas definiert werden. Das Prinzip des Mechanismus ist in beiden Fällen dasselbe, doch unterscheidet er sich in einigen weiter unten beschriebenen Details.

DTDs

Kataloge dienen normalerweise dazu, einen Aufruf von einer DTD an eine lokale URI umzuleiten. Dies geschieht in der Katalogdatei durch Mappen von Public und System Identifiern auf die gewünschte lokale URI. Wenn also die `DOCTYPE`-Deklaration in einer XML-Datei gelesen wird, findet ihr Public oder System Identifier über das Katalogdatei-Mapping die gewünschte lokale Ressource.

Für gebräuchliche Schemas ist der **PUBLIC** Identifier normalerweise vordefiniert, sodass für die URI in der Katalogdatei nur der **PUBLIC** Identifier auf den korrekten lokalen Identifier gemappt werden muss. Wenn das XML-Dokument geparst wird, wird der **PUBLIC** Identifier darin gelesen. Wenn dieser Identifier in einer Katalogdatei gefunden wird, wird die entsprechende URL in der Katalogdatei nachgeschlagen und das Schema wird von dort aus gelesen. Wenn also die folgende SVG-Datei in RaptorXML Server geöffnet wird:

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">

<svg width="20" height="20" xml:space="preserve">
  <g style="fill:red; stroke:#000000">
     <rect x="0" y="0" width="15" height="15"/>
     <rect x="5" y="5" width="15" height="15"/>
  </g>
</svg>
```

Der Katalog wird nach dem **PUBLIC** Identifier dieser SVG-Datei durchsucht. Angenommen, die Katalogdatei enthält den folgenden Eintrag:

```
<catalog>
  ...
    <public publicId="-//W3C//DTD SVG 1.1//EN" uri="schemas/svg/svg11.dtd"/>
  ...
</catalog>
```

In diesem Fall gibt es einen Treffer für den **PUBLIC** Identifier, sodass der Lookup-Mechanismus für die SVG DTD an die URL **schemas/svg/svg11.dtd** umgeleitet wird; (dieser Pfad ist relativ zur Katalogdatei). Diese lokale Datei wird dann als DTD für die SVG-Datei verwendet. Wenn im Katalog kein passender Treffer für den **Public** Identifier gefunden wird, wird die URL im XML-Dokument verwendet (im SVG Beispiel oben ist dies die Internet URL **http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd**).

XML-Schemas

In RaptorXML Server können Sie eine **Umleitung zu einem XML-Schema** auch mit Hilfe von Katalogen durchführen. In der XML-Instanzdatei erfolgt die Referenz zum Schema im **xsi:schemaLocation** Attribut des Elements der obersten Ebene des XML-Dokuments. Beispiel:

```
xsi:schemaLocation="http://www.xmlspy.com/schemas/orgchart OrgChart.xsd"
```

Der Wert des **xsi:schemaLocation**-Attributs besteht aus zwei Teilen: einem Namespace-Teil (oben grün) und einem URI-Teil (markiert). Anhand des Namespace-Teils erfolgt im Katalog das Mapping auf die alternative Ressource. So erfolgt etwa im folgenden Katalogeintrag eine Umleitung der Schemareferenz oben auf ein Schema unter einem anderen Pfad.

```
<uri name="http://www.xmlspy.com/schemas/orgchart" uri="C:\MySchemas\OrgChart.xsd"/>
```

Normalerweise ist der URI-Teil des Attributwerts von **xsi:schemaLocation** ein Pfad zum aktuellen Schema. Wenn das Schema jedoch über einen Katalog referenziert wird, muss der URI-Teil nicht auf ein tatsächliches XML-Schema verweisen, muss aber vorhanden sein, damit das Attribut **xsi:schemaLocation** lexikalisch gültig ist. So würde z.B. der Wert `foo` als URI-Teil des Attributwerts genügen, um gültig zu sein.

4.1.2 Katalogstruktur in RaptorXML Server

RaptorXML Server lädt beim Start eine Datei namens **RootCatalog.xml**(*Struktur siehe unten*), die eine Liste von Katalogdateien enthält, die durchsucht werden. Sie können diese Datei bearbeiten und beliebig viele Katalogdateien definieren, die durchsucht werden sollen. Jede davon wird in einem **nextCatalog** referenziert. Diese Katalogdateien werden durchsucht und die URIs darin werden entsprechend ihren Mappings aufgelöst.

Codefragment von RootCatalog.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<catalog xmlns="urn:oasis:names:tc:entity:xmlns:xml:catalog"
   xmlns:spy="http://www.altova.com/catalog_ext"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="urn:oasis:names:tc:entity:xmlns:xml:catalog Catalog.xsd">
  <nextCatalog catalog="%PersonalFolder%/Altova/%AppAndVersionName%/CustomCatalog.xml"/>
  <!-- Include all catalogs under common schemas folder on the first directory level -->
  <nextCatalog spy:recurseFrom="%CommonSchemasFolder%" catalog="catalog.xml"
spy:depth="1"/>
  <nextCatalog spy:recurseFrom="%ApplicationWritableDataFolder%/pkgs/.cache"
catalog="remapping.xml" spy:depth="0"/>
  <nextCatalog catalog="CoreCatalog.xml"/>
</catalog>
```

Im obigen Codefragment wird ein benutzerdefinierter Katalog namens **CustomCatalog.xml** und eine Gruppe von Katalogen, die gebräuchliche Schemas (wie z.B. W3C XML-Schemas udn das SVG-Schema) referenzieren, referenziert.

- **CustomCatalog.xml** befindet sich im Unterordner **etc** des RaptorXML Server-Applikationsordners. Sie müssen die Datei anhand einer Vorlagendatei namens **CustomCatalog_template.xml** erstellen. Diese Datei ist eine Skelettdatei, in der Sie Ihre eigenen Mappings erstellen können. Sie können für jedes gewünschte Schema, das nicht in den Katalogdateien im Altova Ordner "Common Schemas" enthalten ist, Mappings zu **CustomCatalog.xml** hinzufügen. Verwenden Sie dazu die unterstützten Elemente des OASIS-Katalogmechanismus (*siehe nächster Abschnitt*).
- Der durch die Variable **%CommonSchemasFolder%** definierte Ordner "Common Schemas" enthält eine Reihe gebräuchlicher Schemas. Innerhalb dieser einzelnen Schema-Ordner befindet sich eine **catalog.xml**-Datei, die Public und/oder System Identifier auf URIs mappt, die auf lokal gespeicherte Kopien des jeweiligen Schemas verweisen.
- **CoreCatalog.xml** befindet sich im RaptorXML Server-Applikationsordner und dient zum Auffinden von Schemas und Stylesheets, die von RaptorXML Server-spezifischen Prozessen wie z.B. StyleVision Power Stylesheets, anhand derer die Altova-Authentic-Ansicht von XML-Dokumenten generiert wird.

Beachten Sie dazu Folgendes:

- Bei einer Neuinstallation derselben Hauptversion (selbe oder unterschiedliche Nebenversion) wird die Vorlagendatei durch eine neue Vorlagendatei ersetzt, wobei **CustomCatalog.xml** jedoch unverändert bleibt.
- Wenn Sie jedoch eine neue Hauptversion über eine vorherige Hauptversion installieren, wird der Ordner der vorherigen Hauptversion zusammen mit seiner **CustomCatalog.xml**-Datei gelöscht. Wenn Sie also **CustomCatalog.xml** weiterhin verwenden möchten, müssen Sie **CustomCatalog.xml** aus dem Ordner der vorherigen Hauptversion in einem anderen Ordner sichern. Nach Installation der neuen Hauptversion können Sie die zuvor gespeicherte Datei **CustomCatalog.xml** in den Ordner **etc** der neuen Hauptversion kopieren und dort je nach Bedarf bearbeiten.

Pfadvariablen
Die in **RootCatalog.xml** verwendeten Variablen (*Codefragment oben*) haben die folgenden Werte:

%PersonalFolder%	Der persönliche Ordner des aktuellen Benutzers, z.B. C:\Benutzer\<Name>\Dokumente
%CommonSchemasFolder%	C:\ProgramData\Altova\Common2025\Schemas
%ApplicationWritableDataFolder%	C:\ProgramData\Altova

Speicherpfad von Katalogdateien und Schemas
Beachten Sie die Pfade der verschiedenen Katalogdateien.

- Die Dateien **RootCatalog.xml**, **CustomCatalog.xml**, **CustomCatalog_template.xml** und **CoreCatalog.xml** befinden sich im RaptorXML Server Applikationsordner.
- Die **catalog.xml**-Dateien befinden sich jeweils in einem eigenen Schemaordner, wobei sich diese Schemaordner innerhalb des Ordners "Common Schemas" befinden:

4.1.3 Anpassen von Katalogen

Wenn Sie Einträge für **CustomCatalog.xml** (oder jede andere Katalogdatei, die von RaptorXML Server gelesen werden soll) erstellen, verwenden Sie nur die folgenden Elemente der OASIS-Katalogspezifikation. Jedes der unten angeführten Elemente wird mit einer Erläuterung der Attributwerte aufgelistet. Eine ausführlichere Beschreibung finden Sie in der XML Catalogs Specification. Beachten Sie, dass jedes Element das Attribut **xml:base**, mit dem die Basis-URI dieses Elements definiert wird, erhalten kann.

- <public publicId="PublicID of Resource" uri="URL of local file"/>
- <**system** **systemId**="SystemID of Resource" **uri**="URL of local file"/>
- <**uri** **name**="filename" **uri**="URL of file identified by filename"/>
- <**rewriteURI** **uriStartString**="StartString of URI to rewrite" **rewritePrefix**="String to replace StartString"/>
- <**rewriteSystem** **systemIdStartString**="StartString of SystemID" **rewritePrefix**="Replacement string to locate resource locally"/>

Beachten Sie die folgenden Punkte:

- In Fällen, in denen es keinen Public Identifier gibt, kann der System Identifier einer URL wie z.B. bei den meisten Stylesheets über das **system** Element direkt auf eine URL gemappt werden.
- Eine URI kann über das uri Element auf eine andere URI gemappt werden.
- Mit Hilfe der Elemente rewriteURI und rewriteSystem kann der Anfangsteil einer URI bzw. eines System Identifiers neu geschrieben werden. Dadurch kann der Anfang eines Dateipfads ersetzt werden, sodass ein anderes Verzeichnis als Ziel gewählt werden kann. Nähere Informationen zu diesen Elementen finden Sie in der XML Catalogs Specification.

Ab Release 2014 entspricht RaptorXML Server weitgehend der XML Catalogs Specification (OASIS Standard V1.1, 7 Oktober 2005). In dieser Spezifikation wird streng zwischen externen Identifier Look-ups (jenen mit einer öffentlichen ID oder einer System-ID) und URI Look-ups (URIs, die keine öffentlichen IDS oder System-IDs sind) getrennt. Namespace URIs müssen daher einfach als URIs - und nicht Public IDs oder System-IDs -

behandelt werden und folglich als URI Look-ups anstelle von externen Identifier Look-ups verwendet werden. In RaptorXML Server Versionen vor Version 2014 wurden Schema Namespace URIs über **<public>** Mappings übersetzt. Ab Version 2014 müssen **<uri>** Mappings verwendet werden.

Vor v2014:
```
<public publicID="http://www.MyMapping.com/ref"
    uri="file:///C:/MyDocs/Catalog/test.xsd"/>
```
Ab V-2014:
```
<uri name="http://www.MyMapping.com/ref"
    uri="file:///C:/MyDocs/Catalog/test.xsd"/>
```

Wie findet RaptorXML Server ein referenziertes Schema

Ein Schema wird in einem XML-Dokument über das Attribut **xsi:scemaLocation** (*siehe unten*) referenziert. Der Wert des **xsi:schemaLocation**-Attributs besteht aus zwei Teilen: einem Namespace-Teil (grün) und einem URI-Teil (markiert).

```
xsi:schemaLocation="http://www.xmlspy.com/schemas/orgchart OrgChart.xsd"
```

Welche Schritte anschließend durchgeführt werden müssen, um ein referenziertes Schema zu finden, hängt von den Validierungsoptionen **--schemalocation-hints** und **--schema-mapping** ab. Im Folgenden wird die Vorgangsweise für jeden einzelnen Wert der beiden Optionen beschrieben:

- **--schemalocation-hints=load-by-schemalocation | load-by-namespace | load-combining-both | ignore**
  Definiert das Verhalten der Attribute **xsi:schemaLocation** und **xsi:noNamespaceSchemaLocation**: ob ein Schema-Dokument geladen werden soll und falls ja, anhand welcher Informationen es gesucht werden soll; (die Standardeinstellung ist **load-by-schemalocation**).
  - **load-by-schemalocation**
    1. Wenn der URI-Teil von **xsi:schemaLocation** in einem Katalog gemappt wurde, wird die erzeugte URI geladen.
    2. URI direkt laden.
  - **load-by-namespace**
    1. Wenn der namespace-Teil von **xsi:schemaLocation** in einem Katalog gemappt wurde, wird die erzeugte URI geladen.
    2. Nichts laden.
  - **load-combining-both**
    1. Wenn der URI-Teil von **xsi:schemaLocation** in einem Katalog gemappt wurde, wird die erzeugte URI geladen.
    2. Wenn der namespace-Teil von **xsi:schemaLocation** in einem Katalog gemappt wurde, wird die erzeugte URI geladen.
    3. URI-Teil direkt laden.

- **--schema-mapping=prefer-schemalocation | prefer-namespace**
  Wenn ein Schema-Dokument sowohl anhand des Schema-Pfads als auch des Namespace gesucht werden soll, wird mit dieser Option festgelegt, welche der beiden Optionen beim Katalog-Lookup Vorrang erhält; (die Standardeinstellung ist **prefer-schemalocation**). Mit Hilfe dieser Option wird die Reihenfolge der ersten beiden Schritte in der **load-combining-both**-Variante oben geändert.

XML-Schema-Spezifikationen

Die XML-Schemaspezifikationsinformationen sind in RaptorXML Server integriert und die Gültigkeit von XML-Schema- (`.xsd`)-Dokumenten wird anhand dieser internen Informationen überprüft. Daher sollte in einem XML-Schema-Dokument kein Verweis auf ein Schema, das die XML-Schema-Spezifikation definiert, vorgenommen werden.

Die Datei **`catalog.xml`** im Ordner `%AltovaCommonSchemasFolder%\Schemas\schema` enthält Referenzen auf DTDs, die ältere XML-Schema-Spezifikationen implementieren. Sie sollten Ihre XML-Schema-Dokumente nicht anhand dieser Schemas validieren. Zweck dieser beiden DTDs ist es einzig und allein, für die Eingabehilfen von RaptorXML Server zu Bearbeitungszwecken Informationen bereitzustellen, falls Sie Dateien gemäß diesen älteren Empfehlungen erstellen wollen.

4.1.4 Variablen für Windows-Systempfade

Shell-Umgebungsvariablen können im `nextCatalog` Element verwendet werden, um den Pfad zu Systemordnern zu definieren (*siehe RootCatalog.xml-Liste oben*). Es werden die folgenden Shell-Umgebungsvariablen unterstützt:

`%PersonalFolder%`	Vollständiger Pfad zum persönlichen Ordner des aktuellen Benutzers, z.B. `C:\Benutzer\<Name>\Dokumente`
`%CommonSchemasFolder%`	`C:\ProgramData\Altova\Common2025\Schemas`
`%ApplicationWritableDataFolder%`	`C:\ProgramData\Altova`
`%AltovaCommonFolder%`	C:\Programme\Altova\Common2025
`%DesktopFolder%`	Vollständiger Pfad des Desktop-Ordners des aktuellen Benutzers.
`%ProgramMenuFolder%`	Vollständiger Pfad zum Ordner 'Programme" des aktuellen Benutzers.
`%StartMenuFolder%`	Vollständiger Pfad zum Startmenüordner des aktuellen Benutzers.
`%StartUpFolder%`	Vollständiger Pfad zum Startordner des aktuellen Benutzers.
`%TemplateFolder%`	Vollständiger Pfad des Template-Ordners des aktuellen Benutzers.
`%AdminToolsFolder%`	Vollständiger Pfad zum Dateisystemverzeichnis, in dem Verwaltungstools des aktuellen Benutzers gespeichert sind.
`%AppDataFolder%`	Vollständiger Pfad zum Ordner "Anwendungsdaten" des aktuellen Benutzers.
`%`	Vollständiger Pfad zum Dateisystem, das die Applikationsdaten aller Benutzer enthält.

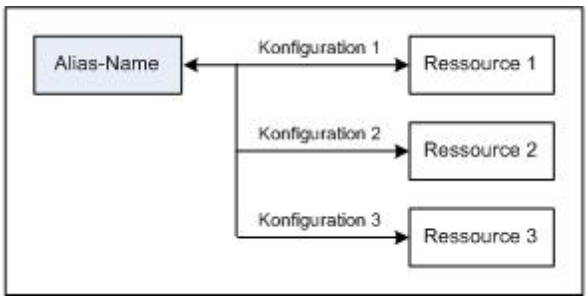
`CommonAppDataFolder%`	
`%FavoritesFolder%`	Vollständiger Pfad zum Ordner 'Favoriten' des aktuellen Benutzers.
`%PersonalFolder%`	Vollständiger Pfad zum Ordner "Personal" des aktuellen Benutzers.
`%SendToFolder%`	Vollständiger Pfad zum SendTo-Ordner des aktuellen Benutzers.
`%FontsFolder%`	Vollständiger Pfad zum Ordner "Systemschriftarten".
`%ProgramFilesFolder%`	Vollständiger Pfad zum Ordner 'Programme" des aktuellen Benutzers.
`%CommonFilesFolder%`	Vollständiger Pfad zum Ordner "Gemeinsame Dateien" des aktuellen Benutzers.
`%WindowsFolder%`	Vollständiger Pfad zum Ordner "Windows" des aktuellen Benutzers.
`%SystemFolder%`	Vollständiger Pfad zum Ordner "System" des aktuellen Benutzers.
`%LocalAppDataFolder%`	Vollständiger Pfad zum Dateisystemverzeichnis, das als Datenspeicher für lokale (nicht-Roaming) Applikationen dient.
`%MyPicturesFolder%`	Vollständiger Pfad zum Ordner "Meine Bilder".

4.2 Globale Ressourcen

In diesem Abschnitt werden folgende Punkte beschrieben:

Informationen zu globalen Ressourcen

In einer Datei für globale Altova-Ressourcen wird ein Alias, wie im Diagramm unten gezeigt, über verschiedene Konfigusrationen auf mehrere Ressourcen gemappt. Um eine andere Ressource aufzurufen, kann daher ein anderer Alias verwendet werden, um die Konfiguration zu wechseln.



Globale Ressourcen werden in Altova-Produkten wie z.B. Altova XMLSpy definiert und in einer XML-Datei für globale Ressourcen gespeichert. RaptorXML kann globale Ressourcen als Inputs verwenden. Dazu werden der Name und Pfad der Datei für globale Ressourcen sowie der Alias und die zu verwendende Konfiguration benötigt.

Der Vorteil der Verwendung globaler Ressourcen ist, dass die Ressource einfach durch Verwendung eines anderen Konfigurationsnamens gewechselt werden kann. Bei Verwendung von RaptorXML kann durch Angabe eines anderen Werts für die `--globalresourcesconfig | --gc` Option eine andere Ressource verwendet werden. (*Siehe Beispiel unten.*)

Verwendung globaler Ressourcen mit RaptorXML

Um eine globale Ressource als Input für einen RaptorXML-Befehl zu verwenden, sind die folgenden Parameter erforderlich:

- Die XML-Datei für globale Ressourcen (in der Befehlszeilenschnittstelle angegeben mit der Option `--globalresourcesfile | --gr`)
- Die erforderliche Konfiguration (in der Befehlszeilenschnittstelle angegeben mit der Option `--globalresourcesconfig | --gc`)
- Der Alias. Dieser kann, wo ein Dateiname erforderlich ist, direkt in der Befehlszeilenschnittstelle angegeben werden oder als Node in einer XML-Datei definiert sein, in dem RaptorXML den Dateinamen abruft (z.B. im Attribut `xsi:schemaLocation`).

Wenn Sie z.B. `input.xml` mittels `transform.xslt` in `output.html` transformieren möchten, erfolgt dies normalerweise über die Befehlszeilenschnittstelle mit dem folgenden Befehl, in dem Dateinamen verwendet werden:

```
raptorxml xslt --input=input.xml --output=output.html transform.xslt
```

Wenn Sie jedoch eine Definition für globale Ressourcen haben, die den Alias `MyInput` über eine Konfigurationsdatei namens `FirstConfig` der Dateiressource `FirstInput.xml` zuordnet, so könnten Sie den Alias `MyInput` in der Befehlszeile folgendermaßen verwenden:

```
raptorxml xslt --input=altova://file_resource/MyInput --gr=C:\MyGlobalResources.xml --
gc=FirstConfig --output=Output.html transform.xslt
```

Angenommen, Sie haben eine andere Dateiressource namens `SecondInput.xml`, die dem Alias `MyInput` über eine Konfiguration namens `SecondConfig` zugeordnet ist, so können Sie diese Ressource verwenden, indem Sie nur die Option `--gc` des vorherigen Befehls ändern:

```
raptorxml xslt --input=altova://file_resource/MyInput --gr=C:\MyGlobalResources.xml --
gc=SecondConfig --output=Output.html transform.xslt
```

Anmerkung: Im obigen Beispiel wurde eine Dateiressource verwendet; einer Dateiressource muss das Präfix **`altova://file_resource/`** vorangestellt werden. Sie können auch globale Ressourcen verwenden, die Ordner sind. Um eine Ordnerressource zu identifizieren, verwenden Sie: **`altova://folder_resource/AliasName`**. Beachten Sie, dass Sie in der Befehlszeilenschnittstelle auch Ordnerressourcen als Teil eines Dateipfads verwenden können. Beispiel: **`altova://folder_resource/AliasName/input.xml`**.

4.3 Sicherheitsfragen

In diesem Abschnitt werden folgende Punkte beschrieben:

Einige Schnittstellenfunktionalitäten von RaptorXML Server können ein Sicherheitsrisiko darstellen. Im Folgenden werden diese sowie die Lösung des Problems beschrieben.

Sicherheitsfragen im Zusammenhang mit der HTTP REST-Schnittstelle

Standardmäßig können Ergebnisdokumente über die HTTP REST-Schnittstelle in jeden durch den Client angegebenen Ordner (auf den über das HTTP-Protokoll Zugriff besteht) geschrieben werden. Beim Konfigurieren von RaptorXML Server sollte dieser Sicherheitsaspekt daher berücksichtigt werden.

Falls die Sicherheit eventuell gefährdet sein könnte oder jemand über die Schnittstelle unbefugten Zugriff erhalten könnte, kann der Server so konfiguriert werden, dass Ergebnisdokumente in ein eigenes Ausgabeverzeichnis auf dem Server selbst geschrieben werden. Dies geschieht durch Setzen der Option `server.unrestricted-filesystem-access` 274 der Server-Konfigurationsdatei auf `false`. Wenn der Zugriff auf diese Weise eingeschränkt ist, kann der Client Ergebnisdokumente aus dem dafür vorgesehenen Ausgabeverzeichnis mit `GET` Requests herunterladen. Alternativ dazu kann ein Administrator die Ergebnisdokumentdateien vom Server in den Zielordner kopieren/laden.

Python-Skripts sicher machen

Wenn ein Python-Skript in einem Befehl über HTTP an RaptorXML Server adressiert ist, funktioniert das Skript nur, wenn es sich im vertrauenswürdigen Verzeichnis 274 befindet. Das Skript wird vom vertrauenswürdigen Verzeichnis aus ausgeführt. Wenn Sie ein Python-Skript aus einem anderen Verzeichnis definieren, wird ein Fehler ausgegeben. Das vertrauenswürdige Verzeichnis wird in der `server.script-root-dir` 273 Einstellung der Serverkonfigurationsdatei 272 definiert. Wenn Sie Python-Skripts verwenden möchten, **muss** ein vertrauenswürdiges Verzeichnis definiert werden. Stellen Sie sicher, dass alle Python-Skripts, die verwendet werden sollen, in diesem Verzeichnis gespeichert werden.

Zwar werden alle vom Server für HTTP-Auftragsanforderungen generierten Ausgabedateien in das Auftragsausgabeverzeichnis 274 (ein Unterverzeichnis von `output-root-directory` 274) geschrieben, doch gilt diese Einschränkung nicht für Python-Skripts, die in jeden Ordner geschrieben werden können. Der Server-Administrator muss die Python-Skripts im vertrauenswürdigen Verzeichnis 274 auf potentielle Schwachstellen überprüfen.

5 Befehlszeilenschnittstelle (CLI)

Die ausführbare RaptorXML-Datei bietet Applikationsfunktionalitäten, die über die Befehlszeilenschnittstelle (CLI) aufgerufen werden können. Der Pfad zur ausführbaren Datei lautet:

Linux	`/opt/Altova/RaptorXMLServer2025/bin/`**`raptorxml`**
Mac	`/usr/local/Altova/RaptorXMLServer2025/bin/`**`raptorxml`**
Windows	`<`*`ProgramFilesFolder`*`>\Altova\RaptorXMLServer2025\bin\`**`RaptorXML.exe`**

Verwendung

Die Befehlszeilensyntax lautet:

```
raptorxml --h | --help | --version | <command> [options] [arguments]
```

- `--help` (Kurzform `--h`) Zeigt den Hilfetext zum jeweiligen Befehl an. Wenn kein Befehl angegeben ist, werden alle Befehle der ausführbaren Datei mit jeweils einer kurzen Beschreibung des Befehls aufgelistet.
- `--version` Zeigt die Versionsnummer von RaptorXML Server an.
- `<command>` ist der auszuführende Befehl. Die Befehle sind in den Unterabschnitten dieses Abschnitts beschrieben (*siehe Liste unten*).
- `[options]` sind die Optionen eines Befehls. Diese werden mit ihren jeweiligen Befehlen aufgelistet und beschrieben.
- `[arguments]` sind die Argumente eines Befehls. Diese werden mit ihren jeweiligen Befehlen aufgelistet und beschrieben.

▼ Groß- und Kleinschreibung und Schrägstriche in der Befehlszeile

`RaptorXML` (und **`RaptorXMLServer`** für Administrator-Befehle) *unter Windows*
`raptorxml` und **`raptorxmlserver`** für Administrator-Befehle) *unter Windows und Unix (Linux, Mac)*

* Beachten Sie, dass klein geschriebene Befehle (`raptorxml` und `raptorxmlserver`) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (`RaptorXML`) nur unter Windows und Mac ausgeführt werden.
*Verwenden Sie auf Linux und Mac-Systemen Schrägstriche und auf Windows-Systemen umgekehrte Schrägstriche.

CLI-Befehle

Im Folgenden sind die CLI-Befehle nach Funktionalität geordnet beschrieben.

5.1 XML-, DTD-, XSD-Validierungsbefehle

Die XML-Validierungsbefehle dienen zum Validieren der folgenden Dokumenttypen:

- valxml-withdtd 60: Validiert ein XML-Instanzdokument anhand einer DTD.
- valxml-withxsd 65: Validiert ein XML-Instanzdokument anhand eines XML-Schemas.
- valdtd 72: Validiert ein DTD-Dokument.
- valxsd 77: Validiert ein W3C XML-Schema-Dokument (XSD).

5.1.1 valxml-withdtd (xml)

Der Befehl **`valxml-withdtd | xml`** validiert ein oder mehrere XML-Dokumente anhand einer DTD.

```
raptorxml valxml-withdtd | xml [options] InputFile
```

- Das Argument *`InputFile`* ist das zu validierende XML-Dokument. Wenn das XML-Dokument eine Referenz auf eine DTD enthält, wird die Option `--dtd` nicht benötigt.
- Um mehrere Input-Dokumente zu validieren, (i) listen Sie entweder die zu validierenden Dateien im CLI auf, wobei jede Datei durch ein Leerzeichen von der nächsten getrennt wird, oder (ii) listen Sie die zu validierenden Dateien in einer Textdatei (`.txt`-Datei) auf (ein Dateiname pro Zeile) und geben Sie diese Textdatei zusammen mit der auf `true` gesetzten Option `--listfile` 252 als das *`InputFile`* Argument an (*siehe Optionsliste unten*).

Beispiele

Beispiele für den Befehl **`valxml-withdtd`**:

- **`raptorxml`** `valxml-withdtd --dtd=c:\MyDTD.dtd c:\Test.xml`
- **`raptorxml`** `xml c:\Test.xml`
- **`raptorxml`** `xml --verbose=true c:\Test.xml`
- **`raptorxml`** `xml --listfile=true c:\FileList.txt`

▼ Groß- und Kleinschreibung und Schrägstriche in der Befehlszeile

`RaptorXML` (und **`RaptorXMLServer`** für Administrator-Befehle) *unter Windows*
`raptorxml` und **`raptorxmlserver`** für Administrator-Befehle) *unter Windows und Unix (Linux, Mac)*

* Beachten Sie, dass klein geschriebene Befehle (`raptorxml` und `raptorxmlserver`) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (`RaptorXML`) nur unter Windows und Mac ausgeführt werden.
*Verwenden Sie auf Linux und Mac-Systemen Schrägstriche und auf Windows-Systemen umgekehrte Schrägstriche.

▼ Groß- und Kleinschreibung und Schrägstriche in der Befehlszeile

`RaptorXML` (und **`RaptorXMLServer`** für Administrator-Befehle) *unter Windows*
`raptorxml` und **`raptorxmlserver`** für Administrator-Befehle) *unter Windows und Unix (Linux, Mac)*

* Beachten Sie, dass klein geschriebene Befehle (`raptorxml` und `raptorxmlserver`) auf allen Plattformen

(Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (`RaptorXML`) nur unter Windows und Mac ausgeführt werden.
*Verwenden Sie auf Linux und Mac-Systemen Schrägstriche und auf Windows-Systemen umgekehrte Schrägstriche.

Optionen

Optionen werden in ihrer kurzen Form (falls verfügbar) und in ihrer langen Form aufgelistet. Für die kurze und die lange Form können ein oder zwei Bindestriche verwendet werden. Eine Option kann, muss aber keinen Wert erhalten. Eine Option, die einen Wert erhält, wird folgendermaßen geschrieben: **`--option=wert`**. Werte können außer in zwei Fällen ohne Anführungszeichen definiert werden: (i) wenn der Wertestring Leerzeichen enthält oder (ii) wenn in der Beschreibung der Option explizit erwähnt ist, dass Anführungszeichen zwingend erforderlich sind. Wenn eine Option einen Booleschen Wert erhält und kein Wert definiert ist, so ist der Standardwert der Option `TRUE`. Mit Hilfe der Option **`--h, --help`** können Sie Informationen über den Befehl anzeigen.

Validierung und Verarbeitung

dtd

`--dtd =` *`FILE`*
Definiert das für die Validierung zu verwendende externe DTD-Dokument. Wenn das XML-Dokument eine Referenz auf eine externe DTD enthält, setzt die CLI-Option die externe Referenz außer Kraft.

listfile

`--listfile =` `true|false`
Bei `true` wird das Argument *`InputFile`* des Befehls als Textdatei behandelt, die einen Dateinamen pro Zeile enthält. Der Standardwert ist `false`. (Als Alternative können die Dateien im CLI getrennt durch ein Leerzeichen aufgelistet werden. Beachten Sie allerdings, dass CLIs eine maximale Zeichenanzahl haben). Beachten Sie, dass die Option `--listfile` nur auf Argumente, nicht aber auf Optionen angewendet wird.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

namespaces

`--namespaces =` `true|false`
Aktiviert die Verarbeitung unter Berücksichtigung des Namespace. Dies ist nützlich, um die XML-Instanz auf Fehler aufgrund falscher Namespaces zu überprüfen. Der Standardwert ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

recurse

`--recurse =` `true|false`
Dient zur Auswahl von Dateien innerhalb von Unterverzeichnissen einschließlich ZIP-Archiven. Bei `true` wählt das Argument *`InputFile`* des Befehls die angegebene Datei auch in den Unterverzeichnissen aus. Beispiel: `"test.zip|zip\test.xml"` wählt Dateien mit dem Namen `test.xml` auf allen Ordnerebenen des ZIP-Ordners aus. Referenzen auf ZIP-Dateien müssen in Anführungszeichen angegeben werden. Es können die Platzhalter `*` und `?` verwendet werden. Mit

`*.xml` werden folglich alle `.xml` Dateien im (ZIP-)Ordner ausgewählt. Der Standardwert der Option ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- script

  `--script =` *`FILE`*
  Führt nach Abschluss der Validierung das Python-Skript in der angegebenen Datei aus. Fügen Sie die Option mehrmals hinzu, um mehr als ein Skript zu definieren.

- script-api-version

  `--api, --script-api-version =` `1; 2; 2.1 to 2.4; 2.4.1; 2.5 bis 2.8; 2.8.1 bis 2.8.6; 2.9.0; 2.10.0; 2.11.0`
  Definiert, welche Python API-Version für das Skript verwendet werden soll. Der Standardwert ist die neueste Version, derzeit **`2.11.0`**. Anstelle von Ganzzahlwerten wie `1` und `2` können Sie auch die entsprechenden Werte `1.0` und `2.0` verwenden. Ebenso können Sie anstelle der zwei Ziffern `2.5` die drei Ziffern `2.5.0` verwenden. Siehe auch Kapitel Python API-Versionen 392.

- script-output

  `--script-output =` *`FILE`*
  Schreibt die Standardausgabe des Skripts in die in *`FILE`* angegebene Datei.

- script-param

  `--script-param =` *`KEY:VALUE`*
  Zusätzliche benutzerdefinierte Parameter, die während der Ausführung von Python Skripts aufgerufen werden können. Fügen Sie die Option mehrmals hinzu, um mehr als einen Parameter zu definieren.

- streaming

  `--streaming =` `true|false`
  Aktiviert die Streaming-Validierung. Standardwert ist `true`. Die im Arbeitsspeicher gehaltene Datenmenge wird im Streaming-Modus minimiert. Der Nachteil ist, dass später eventuell benötigte Informationen - z.B. ein Datenmodell des XML-Instanzdokuments - nicht mehr verfügbar sind. In Situationen, in denen dies eine Rolle spielt, muss der Streaming-Modus deaktiviert werden (indem Sie `--streaming` auf den Wert `false` setzen). Wenn Sie die Option `--script` mit dem Befehl `valxml-withxsd` verwenden, sollten Sie das Streaming deaktivieren. Beachten Sie, dass die Option `--streaming` ignoriert wird, wenn `--parallel-assessment` auf `true` gesetzt wird.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- Kataloge und globale Ressourcen

  - catalog

    `--catalog =` *`FILE`*
    Gibt den absoluten Pfad zu einer Root-Katalog-Datei an, die nicht die installierte Root-Katalog-Datei ist. Der Standardwert ist der absolute Pfad zur installierten Root-Katalog-Datei. (`<installationsordner>\Altova\RaptorXMLServer2025\etc\RootCatalog.xml`). Informationen

zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge[48].

- user-catalog

  --user-catalog = *FILE*
  Definiert den absoluten Pfad zu einem XML-Katalog, der zusätzlich zum Root-Katalog verwendet werden soll. Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge[48].

- enable-globalresources

  --enable-globalresources = `true|false`
  Aktiviert die globalen Ressourcen[55]. Standardwert ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- globalresourceconfig [gc]

  --gc | --globalresourceconfig = *VALUE*
  Definiert die aktive Konfiguration der globalen Ressource[55] (und aktiviert globale Ressourcen)[55].

- globalresourcefile [gr]

  --gr | --globalresourcefile = *FILE*
  Definiert die globale Ressourcendatei[55] (und aktiviert globale Ressourcen)[55].

- Allgemeine Optionen
  - error-format

    --error-format = `text|shortxml|longxml`
    Definiert das Format der Fehlerausgabe. Der Standardwert ist `text`. Mit den anderen Optionen werden XML-Formate generiert, wobei mit `longxml` mehr Details generiert werden.

  - error-limit

    --error-limit = *N | unlimited*
    Definiert das Fehlerlimit mit einem Wertebereich von `1` bis `9999` oder `unbegrenzt`. Der Standardwert ist `100`. Bei Erreichung des Fehlerlimits wird die Validierung gestoppt. Dient dazu, die Prozessorverwendung während der Validierung/Transformation einzuschränken.

  - info-limit

    --info-limit = *N | unlimited*
    Definiert die Grenze für Informationsmeldungen im Bereich von `1-65535` oder als `unlimited` (unbegrenzt). Die Verarbeitung wird auch nach Erreichen des Info-Limits fortgesetzt, doch werden weitere Meldungen nicht mehr ausgegeben. Der Standardwert ist `100`.

  - help

    --help
    Zeigt den Hilfetext zum Befehl an. Beispiel: `valany --h`. (Alternativ dazu kann der Befehl `help` zusammen mit einem Argument verwendet werden. Beispiel: `help valany`.)

- listfile

  --listfile = `true|false`
  Bei `true` wird das Argument *`InputFile`* des Befehls als Textdatei behandelt, die einen Dateinamen pro Zeile enthält. Der Standardwert ist `false`. (Als Alternative können die Dateien im CLI getrennt durch ein Leerzeichen aufgelistet werden. Beachten Sie allerdings, dass CLIs eine maximale Zeichenanzahl haben). Beachten Sie, dass die Option `--listfile` nur auf Argumente, nicht aber auf Optionen angewendet wird.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- log-output

  --log-output = *`FILE`*
  Schreibt die Meldungsausgabe in die angegebene URL. Stellen Sie sicher, dass das CLI Schreibrechte für den Ausgabepfad hat.

- network-timeout

  --network-timeout = *`VALUE`*
  Definiert das Timeout für entfernte I/O-Operationen in Millisekunden. Der Standardwert ist: `40000`.

- recurse

  --recurse = `true|false`
  Dient zur Auswahl von Dateien innerhalb von Unterverzeichnissen einschließlich ZIP-Archiven. Bei `true` wählt das Argument *`InputFile`* des Befehls die angegebene Datei auch in den Unterverzeichnissen aus. Beispiel: `"test.zip|zip\test.xml"` wählt Dateien mit dem Namen `test.xml` auf allen Ordnerebenen des ZIP-Ordners aus. Referenzen auf ZIP-Dateien müssen in Anführungszeichen angegeben werden. Es können die Platzhalter `*` und `?` verwendet werden. Mit `*.xml` werden folglich alle `.xml` Dateien im (ZIP-)Ordner ausgewählt. Der Standardwert der Option ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- verbose

  --verbose = `true|false`
  Mit dem Wert `true` wird die Ausgabe zusätzlicher Informationen bei der Validierung aktiviert. Der Standardwert ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- verbose-output

  --verbose-output = *`FILE`*
  Schreibt die ausführliche Ausgabe in *`FILE`*.

- version

  --version
  Zeigt die Version von RaptorXML Server an. Setzen Sie `--version` bei Verwendung mit einem Befehl

vor den Befehl.

- warning-limit

  --warning-limit = *N | unlimited*
  Definiert das Warnungslimit im Bereich von `1-65535` oder `unbegrenzt`. Bei Erreichen dieses Limits wird die Verarbeitung fortgesetzt, doch werden keine weiteren Warnungen mehr ausgegeben. Der Standardwert ist `100`.

5.1.2 valxml-withxsd (xsi)

Der Befehl **`valxml-withxsd | xsi`** validiert ein oder mehrere XML-Instanzdokumente anhand der W3C XML Schema Definition Language (XSD) Spezifikationen 1.0 und 1.1.

```
raptorxml valxml-withxsd | xsi [options] InputFile
```

- Das Argument *`InputFile`* ist das zu validierende XML-Dokument. Die Option `--schemalocation-hints`[255] gibt an, mit Hilfe welcher Mechanismen das Schema gefunden werden soll. Mit der Option `--xsd=FILE`[253] wird das/die zu verwendende(n) Schema(s) angegeben, wenn die XML-Datei keine Schemareferenz enthält.
- Um mehrere Dokumente zu validieren, (i) listen Sie entweder die zu validierenden Dateien im CLI auf, wobei jede Datei durch ein Leerzeichen von der nächsten getrennt wird, oder (ii) listen Sie die zu validierenden Dateien in einer Textdatei (`.txt`-Datei) auf (ein Dateiname pro Zeile) und geben Sie diese Textdatei zusammen mit der auf `true` gesetzten Option `--listfile`[252] als das *`InputFile`* Argument an (*siehe Optionsliste unten*).

Anmerkung: Wenn Sie zum Ausführen von Python-Skripts[391] die Option `--script` verwenden, stellen Sie sicher, dass Sie auch `--streaming=false` definieren.

Beispiele

Beispiele für den Befehl **`valxml-withxsd`**:

- **`raptorxml`** `valxml-withxsd --schemalocation-hints=load-by-schemalocation --xsd=c:\MyXSD.xsd c:\HasNoXSDRef.xml`
- **`raptorxml`** `xsi c:\HasXSDRef.xml`
- **`raptorxml`** `xsi --xsd-version=1.1 --listfile=true c:\FileList.txt`

- Groß- und Kleinschreibung und Schrägstriche in der Befehlszeile

  `RaptorXML` (und **`RaptorXMLServer`** für Administrator-Befehle) *unter Windows*
  `raptorxml` und **`raptorxmlserver`** für Administrator-Befehle) *unter Windows und Unix (Linux, Mac)*

  * Beachten Sie, dass klein geschriebene Befehle (`raptorxml` und `raptorxmlserver`) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (`RaptorXML`) nur unter Windows und Mac ausgeführt werden.
  *Verwenden Sie auf Linux und Mac-Systemen Schrägstriche und auf Windows-Systemen umgekehrte

Schrägstriche.

- Umgekehrte Schrägstriche, Leerzeichen und Sonderzeichen auf Windows-Systemen

  Auf Windows-Systemen: Verwenden Sie bei Vorhandensein von Leerzeichen oder Sonerzeichen in Strings (z.B. in Datei- oder Ordnernamen oder Firmen-, Personen- oder Produktnamen) Anführungszeichen: z.B., **"Meine Datei"**. Beachten Sie jedoch, dass ein von einem doppelten Anführungszeichen gefolgter umgekehrter Schrägstrich (z.B: **"C:\Mein Verzeichnis\"**) eventuell nicht korrekt gelesen wird, da der umgekehrte Schrägstrich auch den Beginn einer Escape-Sequenz markiert und die Escape-Sequenz **\"** für ein doppeltes Anführungszeichen steht. Wenn Sie diese Zeichensequenz mit einem Escape versehen wollen, verwenden Sie einen vorangestellten umgekehrten Schrägstrich, wie den folgenden: **\\"**. Einfacher ausgedrückt: Wenn Sie einen Dateipfad, der Leerzeichen oder einen umgekehrten Schrägstrich am Ende enthält, schreiben müssen, so schreiben Sie diesen folgendermaßen: **"C:\Mein Verzeichnis\\"**.

Optionen

Optionen werden in ihrer kurzen Form (falls verfügbar) und in ihrer langen Form aufgelistet. Für die kurze und die lange Form können ein oder zwei Bindestriche verwendet werden. Eine Option kann, muss aber keinen Wert erhalten. Eine Option, die einen Wert erhält, wird folgendermaßen geschrieben: **--option=wert**. Werte können außer in zwei Fällen ohne Anführungszeichen definiert werden: (i) wenn der Wertestring Leerzeichen enthält oder (ii) wenn in der Beschreibung der Option explizit erwähnt ist, dass Anführungszeichen zwingend erforderlich sind. Wenn eine Option einen Booleschen Wert erhält und kein Wert definiert ist, so ist der Standardwert der Option `TRUE`. Mit Hilfe der Option **--h, --help** können Sie Informationen über den Befehl anzeigen.

- Validierung ung Verarbeitung
  - assessment-mode

    --assessment-mode = `lax|strict`
    Definiert den Beurteilungsmodus für die Gültigkeit von Schemas gemäß der XSD-Spezifikation. Der Standardwert ist `strict`. Das XML-Instanzdokument wird entsprechend dem mit dieser Option definierten Modus validiert.

  - ct-restrict-mode

    --ct-restrict-mode = `1.0|1.1|default`
    Definiert, wie complexType-Einschränkungen überprüft werden sollen. Beim Wert **1.0** werden complexType-Einschränkungen anhand der XSD 1.0-Spezifikation überprüft - und zwar auch im XSD 1.1-Validierungsmodus. Beim Wert **1.1** werden complexType-Einschränkungen anhand der XSD 1.1-Spezifikation überprüft - und zwar auch im XSD 1.0-Validierungsmodus. Beim Wert **default** werden complexType-Einschränkungen anhand der als aktueller Validierungsmodus (1.0 oder 1.1) ausgewählten XSD-Spezifikation überprüft. Der Standardwert ist **default**.

  - listfile

    --listfile = `true|false`
    Bei `true` wird das Argument *InputFile* des Befehls als Textdatei behandelt, die einen Dateinamen pro Zeile enthält. Der Standardwert ist `false`. (Als Alternative können die Dateien im CLI getrennt durch ein Leerzeichen aufgelistet werden. Beachten Sie allerdings, dass CLIs eine maximale Zeichenanzahl haben). Beachten Sie, dass die Option `--listfile` nur auf Argumente, nicht aber auf Optionen angewendet wird.

Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- parallel-assessment [pa]

  `--pa | --parallel-assessment =` `true|false`
  Bei Setzung auf `true` wird die Schema-Validierung parallel ausgeführt. Das bedeutet, wenn sich auf irgendeiner Ebene mehr als 128 Elemente befinden, so werden diese Elemente über mehrere Threads parallel verarbeitet. Auf diese Weise können besonders große XML-Dateien schneller verarbeitet werden, wenn diese Option aktiv ist. Parallele Validierungen können gleichzeitig auf einer hierarchischen Ebene ausgeführt werden, können in einem einzigen Infoset aber auch auf mehreren Ebenen erfolgen. Beachten Sie dass die parallele Validierung im Streaming-Modus nicht funktioniert. Aus diesem Grund wird die Option `--streaming` ignoriert, wenn `--parallel-assessment` auf `true`. gesetzt ist. Außerdem wird bei Verwendung der Option `--parallel-assessment` mehr Arbeitsspeicher benötigt. Die Standardeinstellung ist `false`. Die Kurzform für die Option ist `--pa`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- recurse

  `--recurse =` `true|false`
  Dient zur Auswahl von Dateien innerhalb von Unterverzeichnissen einschließlich ZIP-Archiven. Bei `true` wählt das Argument *`InputFile`* des Befehls die angegebene Datei auch in den Unterverzeichnissen aus. Beispiel: `"test.zip|zip\test.xml"` wählt Dateien mit dem Namen `test.xml` auf allen Ordnerebenen des ZIP-Ordners aus. Referenzen auf ZIP-Dateien müssen in Anführungszeichen angegeben werden. Es können die Platzhalter `*` und `?` verwendet werden. Mit `*.xml` werden folglich alle `.xml` Dateien im (ZIP-)Ordner ausgewählt. Der Standardwert der Option ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- report-import-namespace-mismatch-as-warning

  `--report-import-namespace-mismatch-as-warning =` `true|false`
  Stuft Fehler, die beim Import von Schemas mit `xs:import` aufgrund eines nicht übereinstimmenden Namespace oder Ziel-Namespace auftreten, von Fehlern auf Warnungen herab. Der Standardwert ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- schema-imports

  `--schema-imports =` `load-by-schemalocation | load-preferring-schemalocation | load-by-namespace | load-combining-both | license-namespace-only`
  Definiert das Verhalten von `xs:import` Elementen, von denen jedes ein optionales `namespace` Attribut und ein optionales `schemaLocation` Attribut hat: `<import namespace="someNS" schemaLocation="someURL">`. Mit der Option wird definiert, ob ein Schema-Dokument geladen oder nur ein Namespace lizenziert werden soll und, wenn ein Schema-Dokument geladen werden soll, anhand welcher Informationen es gesucht werden soll. Standardeinstellung: `load-preferring-schemalocation`. Das Verhalten ist das folgende:

  - `load-by-schemalocation`: Anhand des Werts des `schemaLocation` Attributs wird der

Schemapfad ermittelt, wobei Katalog-Mappings[48] berücksichtigt werden. Wenn das Namespace-Attribut vorhanden ist, wird der Namespace importiert (lizenziert).

- `load-preferring-schemalocation`: Wenn das `schemaLocation` Attribut vorhanden ist, wird es verwendet, wobei Katalog-Mappings[48] berücksichtigt werden. Falls kein `schemaLocation` Attribut vorhanden ist, wird der Wert des `namespace` Attributs über ein Katalog-Mapping[48] verwendet. Dies ist der **Standardwert**.
- `load-by-namespace`: Anhand des Werts des `namespace` Attributs wird der Schemapfad über ein Katalog-Mapping[48] ermittelt.
- `load-combining-both`: Wenn entweder das Attribut `namespace` oder das Attribut `schemaLocation` ein Katalog-Mapping[48] hat, so wird das Mapping verwendet. Wenn beide Attribute Katalog-Mappings[48] haben, ist es vom Wert der Option `--schema-mapping` (XML/XSD-Option[255]) abhängig, welches Mapping verwendet wird. Falls kein Katalog-Mapping[48] vorhanden ist, wird das `schemaLocation` Attribut verwendet.
- `license-namespace-only`: Der Namespace wird importiert. Kein Schema-Dokument wird importiert.

▼ schema-location-hints

`--schema-location-hints =` `load-by-schemalocation | load-by-namespace | load-combining-both | ignore`
Definiert das Verhalten der Attribute `xsi:schemaLocation` und `xsi:noNamespaceSchemaLocation`: Legt fest, ob ein Schema-Dokument geladen werden soll und falls ja, anhand welcher Informationen es gesucht werden soll. Standardeinstellung: `load-by-schemalocation`.

- Der Wert `load-by-schemalocation` verwendet die URL des Schemapfads[422] in den Attributen `xsi:schemaLocation` und `xsi:noNamespaceSchemaLocation` in XML-Instanzdokumenten. Dies ist der **Standardwert**.
- Der Wert `load-by-namespace` verwendet den Namespace-Teil[422] von `xsi:schemaLocation` und - im Fall von `xsi:noNamespaceSchemaLocation` - einen leeren String und ermittelt das Schema über ein Katalog-Mapping[48].
- Bei Verwendung von `load-combining-both` und wenn entweder der Namespace-Teil oder der URL-Teil ein Katalog-Mapping[48] hat, so wird das Katalog-Mapping[48] verwendet. Wenn beide Katalog-Mappings[48] haben, ist es vom Wert der `--schema-mapping` Option (XML/XSD-Option[255]) abhängig, welches Mapping verwendet wird. Wenn weder der Namespace noch die URL ein Katalog-Mapping hat, wird die URL verwendet.
- Wenn der Wert der Option `ignore` ist, werden die beiden Attribute `xsi:schemaLocation` und `xsi:noNamespaceSchemaLocation` ignoriert.

▼ schema-mapping

`--schema-mapping =` `prefer-schemalocation | prefer-namespace`
Wenn ein Schema-Dokument sowohl anhand des Schema-Pfads als auch des Namespace gesucht werden soll, wird damit festgelegt, welche der beiden Optionen beim Katalog-Lookup Vorrang erhält. (Wenn eine der Optionen `--schemalocation-hints` oder `--schema-imports` einen Wert `load-combining-both` hat und wenn die betroffenen Namespace- und URL-Teile beide Katalog-Mappings[48] haben, gibt der Wert dieser Option an, welches der beiden Mappings verwendet werden soll (das Namespace Mapping oder das URL-Mapping; der Wert `prefer-schemalocation` bezieht sich auf das URL-Mapping)). Der Standardwert ist `prefer-schemalocation`.

▼ script

`--script =` *`FILE`*
Führt nach Abschluss der Validierung das Python-Skript in der angegebenen Datei aus. Fügen Sie

die Option mehrmals hinzu, um mehr als ein Skript zu definieren.

- script-api-version

  `--api, --script-api-version =` `1; 2; 2.1 to 2.4; 2.4.1; 2.5 bis 2.8; 2.8.1 bis 2.8.6; 2.9.0; 2.10.0; 2.11.0`
  Definiert, welche Python API-Version für das Skript verwendet werden soll. Der Standardwert ist die neueste Version, derzeit **`2.11.0`**. Anstelle von Ganzzahlwerten wie `1` und `2` können Sie auch die entsprechenden Werte `1.0` und `2.0` verwenden. Ebenso können Sie anstelle der zwei Ziffern `2.5` die drei Ziffern `2.5.0` verwenden. Siehe auch Kapitel Python API-Versionen 392.

- script-output

  `--script-output =` *`FILE`*
  Schreibt die Standardausgabe des Skripts in die in *`FILE`* angegebene Datei.

- script-param

  `--script-param =` *`KEY:VALUE`*
  Zusätzliche benutzerdefinierte Parameter, die während der Ausführung von Python Skripts aufgerufen werden können. Fügen Sie die Option mehrmals hinzu, um mehr als einen Parameter zu definieren.

- streaming

  `--streaming =` `true|false`
  Aktiviert die Streaming-Validierung. Standardwert ist `true`. Die im Arbeitsspeicher gehaltene Datenmenge wird im Streaming-Modus minimiert. Der Nachteil ist, dass später eventuell benötigte Informationen - z.B. ein Datenmodell des XML-Instanzdokuments - nicht mehr verfügbar sind. In Situationen, in denen dies eine Rolle spielt, muss der Streaming-Modus deaktiviert werden (indem Sie `--streaming` auf den Wert `false` setzen). Wenn Sie die Option `--script` mit dem Befehl `valxml-withxsd` verwenden, sollten Sie das Streaming deaktivieren. Beachten Sie, dass die Option `--streaming` ignoriert wird, wenn `--parallel-assessment` auf `true` gesetzt wird.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- xinclude

  `--xinclude =` `true|false`
  Aktiviert die Unterstützung für XML-Inkludierungen (XInclude). Der Standardwert ist `false`. Bei `false` werden die `include`-Elemente von XInclude ignoriert.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- xml-mode

  `--xml-mode =` `wf|id|valid`
  Definiert, welcher XML-Verarbeitungsmodus für das XML-Instanzdokument verwendet werden soll: `wf`=Wohlgeformtheitsprüfung; `id`=Wohlgeformtheitsprüfung mit ID/IDREF-Prüfung; `valid`=Validierung. Der Standardwert ist `wf`. Beachten Sie, dass jedes bei der Verarbeitung geladene Instanzdokument eine DTD referenzieren muss, damit der Wert `valid` ausgegeben werden kann. Falls keine DTD vorhanden ist, wird ein Fehler ausgegeben.

- xml-mode-for-schemas

--xml-mode-for-schemas = wf|id|valid
Definiert, welcher XML-Verarbeitungsmodus für XML-Schema-Dokumente verwendet werden soll: wf=Wohlgeformtheitsprüfung; id=Wohlgeformtheitsprüfung mit ID/IDREF-Prüfung; valid=Validierung. Der Standardwert ist wf. Beachten Sie, dass jedes bei der Verarbeitung geladene Schema-Dokument eine DTD referenzieren muss, damit der Wert valid ausgegeben werden kann. Falls keine DTD vorhanden ist, wird ein Fehler ausgegeben.

- xsd

--xsd = *FILE*
Definiert ein oder mehrere XML-Schema-Dokumente, die für die Validierung von XML-Instanzdokumenten verwendet werden sollen. Um mehr als ein Schema-Dokument zu definieren, fügen Sie die Option mehrmals hinzu.

- xsd-version

--xsd-version = 1.0|1.1|detect
Definiert die zu verwendende W3C Schema Definition Language (XSD) Version. Der Standardwert ist 1.0. Diese Option eignet sich auch, um herauszufinden, inwiefern ein 1.0-kompatibles Schema nicht mit Schemaversion 1.1 kompatibel ist. Die Option detect ist eine Altova-spezifische Funktionalität. Mit dieser Option kann die Version des XML-Schema-Dokuments (1.0 oder 1.1) durch Lesen des Werts des vc:minVersion Attributs des <xs:schema> Elements des Dokuments ermittelt werden. Wenn der Wert des @vc:minVersion Attributs 1.1 ist, wird das Schema als Version 1.1 erkannt. Bei jedem anderen Wert wird das Schema als 1.0 erkannt. Bei jedem anderen Wert oder bei Fehlen des @vc:minVersion Attributs wird das Schema als Version 1.0 gelesen.

- Kataloge und globale Ressourcen
  - catalog

--catalog = *FILE*
Gibt den absoluten Pfad zu einer Root-Katalog-Datei an, die nicht die installierte Root-Katalog-Datei ist. Der Standardwert ist der absolute Pfad zur installierten Root-Katalog-Datei. (<installationsordner>\Altova\RaptorXMLServer2025\etc\RootCatalog.xml). Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge 48.

  - user-catalog

--user-catalog = *FILE*
Definiert den absoluten Pfad zu einem XML-Katalog, der zusätzlich zum Root-Katalog verwendet werden soll. Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge 48.

  - enable-globalresources

--enable-globalresources = true|false
Aktiviert die globalen Ressourcen 55. Standardwert ist false.
Hinweis: Die Booleschen Optionswerte werden auf true gesetzt, wenn die Option ohne einen Wert definiert wird.

  - globalresourceconfig [gc]

--gc | --globalresourceconfig = *VALUE*
Definiert die aktive Konfiguration der globalen Ressource 55 (und aktiviert globale Ressourcen) 55.

- globalresourcefile [gr]

  `--gr | --globalresourcefile =` *`FILE`*
  Definiert die globale Ressourcendatei[55] (und aktiviert globale Ressourcen)[55].

- Allgemeine Optionen
  - error-format

    `--error-format =` `text|shortxml|longxml`
    Definiert das Format der Fehlerausgabe. Der Standardwert ist `text`. Mit den anderen Optionen werden XML-Formate generiert, wobei mit `longxml` mehr Details generiert werden.

  - error-limit

    `--error-limit =` *`N | unlimited`*
    Definiert das Fehlerlimit mit einem Wertebereich von `1` bis `9999` oder `unbegrenzt`. Der Standardwert ist `100`. Bei Erreichung des Fehlerlimits wird die Validierung gestoppt. Dient dazu, die Prozessorverwendung während der Validierung/Transformation einzuschränken.

  - info-limit

    `--info-limit =` *`N | unlimited`*
    Definiert die Grenze für Informationsmeldungen im Bereich von `1-65535` oder als `unlimited` (unbegrenzt). Die Verarbeitung wird auch nach Erreichen des Info-Limits fortgesetzt, doch werden weitere Meldungen nicht mehr ausgegeben. Der Standardwert ist `100`.

  - help

    `--help`
    Zeigt den Hilfetext zum Befehl an. Beispiel: `valany --h`. (Alternativ dazu kann der Befehl `help` zusammen mit einem Argument verwendet werden. Beispiel: `help valany`.)

  - listfile

    `--listfile =` `true|false`
    Bei `true` wird das Argument *`InputFile`* des Befehls als Textdatei behandelt, die einen Dateinamen pro Zeile enthält. Der Standardwert ist `false`. (Als Alternative können die Dateien im CLI getrennt durch ein Leerzeichen aufgelistet werden. Beachten Sie allerdings, dass CLIs eine maximale Zeichenanzahl haben). Beachten Sie, dass die Option `--listfile` nur auf Argumente, nicht aber auf Optionen angewendet wird.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

  - log-output

    `--log-output =` *`FILE`*
    Schreibt die Meldungsausgabe in die angegebene URL. Stellen Sie sicher, dass das CLI Schreibrechte für den Ausgabepfad hat.

  - network-timeout

    `--network-timeout =` *`VALUE`*

Definiert das Timeout für entfernte I/O-Operationen in Millisekunden. Der Standardwert ist: `40000`.

- recurse

  `--recurse =` `true|false`
  Dient zur Auswahl von Dateien innerhalb von Unterverzeichnissen einschließlich ZIP-Archiven. Bei `true` wählt das Argument *`InputFile`* des Befehls die angegebene Datei auch in den Unterverzeichnissen aus. Beispiel: `"test.zip|zip\test.xml"` wählt Dateien mit dem Namen `test.xml` auf allen Ordnerebenen des ZIP-Ordners aus. Referenzen auf ZIP-Dateien müssen in Anführungszeichen angegeben werden. Es können die Platzhalter `*` und `?` verwendet werden. Mit `*.xml` werden folglich alle `.xml` Dateien im (ZIP-)Ordner ausgewählt. Der Standardwert der Option ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- verbose

  `--verbose =` `true|false`
  Mit dem Wert `true` wird die Ausgabe zusätzlicher Informationen bei der Validierung aktiviert. Der Standardwert ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- verbose-output

  `--verbose-output =` *`FILE`*
  Schreibt die ausführliche Ausgabe in *`FILE`*.

- version

  `--version`
  Zeigt die Version von RaptorXML Server an. Setzen Sie `--version` bei Verwendung mit einem Befehl vor den Befehl.

- warning-limit

  `--warning-limit =` *`N | unlimited`*
  Definiert das Warnungslimit im Bereich von `1-65535` oder `unbegrenzt`. Bei Erreichen dieses Limits wird die Verarbeitung fortgesetzt, doch werden keine weiteren Warnungen mehr ausgegeben. Der Standardwert ist `100`.

5.1.3 valdtd (dtd)

Der Befehl **`valdtd | dtd`** validiert ein oder mehrere DTD-Dokumente anhand der XML 1.0 oder 1.1-Spezifikation.

```
raptorxml valdtd | dtd [options] InputFile
```

- Das Argument *InputFile* ist das zu validierende DTD-Dokument.
- Um mehrere Dokumente zu validieren, (i) listen Sie entweder die zu validierenden Dateien im CLI auf, wobei jede Datei durch ein Leerzeichen von der nächsten getrennt wird, oder (ii) listen Sie die zu validierenden Dateien in einer Textdatei (`.txt`-Datei) auf (ein Dateiname pro Zeile) und geben Sie diese Textdatei zusammen mit der auf `true` gesetzten Option `--listfile`[252] als das *InputFile* Argument an (*siehe Optionsliste unten*).

Beispiele

Beispiele für den Befehl **valdtd**:

- `raptorxml valdtd c:\Test.dtd`
- **raptorxml** `dtd --verbose=true c:\Test.dtd`
- **raptorxml** `dtd --listfile=true c:\FileList.txt`

▼ Groß- und Kleinschreibung und Schrägstriche in der Befehlszeile

RaptorXML (und **RaptorXMLServer** für Administrator-Befehle) *unter Windows*
raptorxml und **raptorxmlserver** für Administrator-Befehle) *unter Windows und Unix (Linux, Mac)*

* Beachten Sie, dass klein geschriebene Befehle (`raptorxml` und `raptorxmlserver`) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (`RaptorXML`) nur unter Windows und Mac ausgeführt werden.
*Verwenden Sie auf Linux und Mac-Systemen Schrägstriche und auf Windows-Systemen umgekehrte Schrägstriche.

▼ Umgekehrte Schrägstriche, Leerzeichen und Sonderzeichen auf Windows-Systemen

Auf Windows-Systemen: Verwenden Sie bei Vorhandensein von Leerzeichen oder Sonerzeichen in Strings (z.B. in Datei- oder Ordnernamen oder Firmen-, Personen- oder Produktnamen) Anführungszeichen: z.B., **`"Meine Datei"`**. Beachten Sie jedoch, dass ein von einem doppelten Anführungszeichen gefolgter umgekehrter Schrägstrich (z.B: **`"C:\Mein Verzeichnis\"`**) eventuell nicht korrekt gelesen wird, da der umgekehrte Schrägstrich auch den Beginn einer Escape-Sequenz markiert und die Escape-Sequenz **`\"`** für ein doppeltes Anführungszeichen steht. Wenn Sie diese Zeichensequenz mit einem Escape versehen wollen, verwenden Sie einen vorangestellten umgekehrten Schrägstrich, wie den folgenden: **`\\"`**. Einfacher ausgedrückt: Wenn Sie einen Dateipfad, der Leerzeichen oder einen umgekehrten Schrägstrich am Ende enthält, schreiben müssen, so schreiben Sie diesen folgendermaßen: **`"C:\Mein Verzeichnis\\"`**.

Optionen

Optionen werden in ihrer kurzen Form (falls verfügbar) und in ihrer langen Form aufgelistet. Für die kurze und die lange Form können ein oder zwei Bindestriche verwendet werden. Eine Option kann, muss aber keinen Wert erhalten. Eine Option, die einen Wert erhält, wird folgendermaßen geschrieben: **`--option=wert`**. Werte können außer in zwei Fällen ohne Anführungszeichen definiert werden: (i) wenn der Wertestring Leerzeichen enthält oder (ii) wenn in der Beschreibung der Option explizit erwähnt ist, dass Anführungszeichen zwingend erforderlich sind. Wenn eine Option einen Booleschen Wert erhält und kein Wert definiert ist, so ist der Standardwert der Option `TRUE`. Mit Hilfe der Option **`--h, --help`** können Sie Informationen über den Befehl anzeigen.

- Validierung und Verarbeitung
  - listfile

    --listfile = `true|false`
    Bei `true` wird das Argument *InputFile* des Befehls als Textdatei behandelt, die einen Dateinamen pro Zeile enthält. Der Standardwert ist `false`. (Als Alternative können die Dateien im CLI getrennt durch ein Leerzeichen aufgelistet werden. Beachten Sie allerdings, dass CLIs eine maximale Zeichenanzahl haben). Beachten Sie, dass die Option `--listfile` nur auf Argumente, nicht aber auf Optionen angewendet wird.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

  - recurse

    --recurse = `true|false`
    Dient zur Auswahl von Dateien innerhalb von Unterverzeichnissen einschließlich ZIP-Archiven. Bei `true` wählt das Argument *InputFile* des Befehls die angegebene Datei auch in den Unterverzeichnissen aus. Beispiel: `"test.zip|zip\test.xml"` wählt Dateien mit dem Namen `test.xml` auf allen Ordnerebenen des ZIP-Ordners aus. Referenzen auf ZIP-Dateien müssen in Anführungszeichen angegeben werden. Es können die Platzhalter `*` und `?` verwendet werden. Mit `*.xml` werden folglich alle `.xml` Dateien im (ZIP-)Ordner ausgewählt. Der Standardwert der Option ist `false`.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

  - script

    --script = *FILE*
    Führt nach Abschluss der Validierung das Python-Skript in der angegebenen Datei aus. Fügen Sie die Option mehrmals hinzu, um mehr als ein Skript zu definieren.

  - script-api-version

    --api, --script-api-version = `1; 2; 2.1 to 2.4; 2.4.1; 2.5 bis 2.8; 2.8.1 bis 2.8.6; 2.9.0; 2.10.0; 2.11.0`
    Definiert, welche Python API-Version für das Skript verwendet werden soll. Der Standardwert ist die neueste Version, derzeit **`2.11.0`**. Anstelle von Ganzzahlwerten wie `1` und `2` können Sie auch die entsprechenden Werte `1.0` und `2.0` verwenden. Ebenso können Sie anstelle der zwei Ziffern `2.5` die drei Ziffern `2.5.0` verwenden. Siehe auch Kapitel Python API-Versionen 392.

  - script-output

    --script-output = *FILE*
    Schreibt die Standardausgabe des Skripts in die in *FILE* angegebene Datei.

  - script-param

    --script-param = *KEY:VALUE*
    Zusätzliche benutzerdefinierte Parameter, die während der Ausführung von Python Skripts aufgerufen werden können. Fügen Sie die Option mehrmals hinzu, um mehr als einen Parameter zu definieren.

- Kataloge und globale Ressourcen

- catalog

  --catalog = *FILE*
  Gibt den absoluten Pfad zu einer Root-Katalog-Datei an, die nicht die installierte Root-Katalog-Datei ist. Der Standardwert ist der absolute Pfad zur installierten Root-Katalog-Datei. (`<installationsordner>\Altova\RaptorXMLServer2025\etc\RootCatalog.xml`). Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge[48].

- user-catalog

  --user-catalog = *FILE*
  Definiert den absoluten Pfad zu einem XML-Katalog, der zusätzlich zum Root-Katalog verwendet werden soll. Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge[48].

- enable-globalresources

  --enable-globalresources = `true|false`
  Aktiviert die globalen Ressourcen[55]. Standardwert ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- globalresourceconfig [gc]

  --gc | --globalresourceconfig = *VALUE*
  Definiert die aktive Konfiguration der globalen Ressource[55] (und aktiviert globale Ressourcen)[55].

- globalresourcefile [gr]

  --gr | --globalresourcefile = *FILE*
  Definiert die globale Ressourcendatei[55] (und aktiviert globale Ressourcen)[55].

- Allgemeine Optionen
  - error-format

    --error-format = `text|shortxml|longxml`
    Definiert das Format der Fehlerausgabe. Der Standardwert ist `text`. Mit den anderen Optionen werden XML-Formate generiert, wobei mit `longxml` mehr Details generiert werden.

  - error-limit

    --error-limit = *N | unlimited*
    Definiert das Fehlerlimit mit einem Wertebereich von `1` bis `9999` oder `unbegrenzt`. Der Standardwert ist `100`. Bei Erreichung des Fehlerlimits wird die Validierung gestoppt. Dient dazu, die Prozessorverwendung während der Validierung/Transformation einzuschränken.

  - info-limit

    --info-limit = *N | unlimited*
    Definiert die Grenze für Informationsmeldungen im Bereich von `1-65535` oder als `unlimited` (unbegrenzt). Die Verarbeitung wird auch nach Erreichen des Info-Limits fortgesetzt, doch werden weitere Meldungen nicht mehr ausgegeben. Der Standardwert ist `100`.

- help

  --help
  Zeigt den Hilfetext zum Befehl an. Beispiel: `valany --h`. (Alternativ dazu kann der Befehl `help` zusammen mit einem Argument verwendet werden. Beispiel: `help valany`.)

- listfile

  --listfile = `true|false`
  Bei `true` wird das Argument *`InputFile`* des Befehls als Textdatei behandelt, die einen Dateinamen pro Zeile enthält. Der Standardwert ist `false`. (Als Alternative können die Dateien im CLI getrennt durch ein Leerzeichen aufgelistet werden. Beachten Sie allerdings, dass CLIs eine maximale Zeichenanzahl haben). Beachten Sie, dass die Option `--listfile` nur auf Argumente, nicht aber auf Optionen angewendet wird.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- log-output

  --log-output = *`FILE`*
  Schreibt die Meldungsausgabe in die angegebene URL. Stellen Sie sicher, dass das CLI Schreibrechte für den Ausgabepfad hat.

- network-timeout

  --network-timeout = *`VALUE`*
  Definiert das Timeout für entfernte I/O-Operationen in Millisekunden. Der Standardwert ist: `40000`.

- recurse

  --recurse = `true|false`
  Dient zur Auswahl von Dateien innerhalb von Unterverzeichnissen einschließlich ZIP-Archiven. Bei `true` wählt das Argument *`InputFile`* des Befehls die angegebene Datei auch in den Unterverzeichnissen aus. Beispiel: `"test.zip|zip\test.xml"` wählt Dateien mit dem Namen `test.xml` auf allen Ordnerebenen des ZIP-Ordners aus. Referenzen auf ZIP-Dateien müssen in Anführungszeichen angegeben werden. Es können die Platzhalter `*` und `?` verwendet werden. Mit `*.xml` werden folglich alle `.xml` Dateien im (ZIP-)Ordner ausgewählt. Der Standardwert der Option ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- verbose

  --verbose = `true|false`
  Mit dem Wert `true` wird die Ausgabe zusätzlicher Informationen bei der Validierung aktiviert. Der Standardwert ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- verbose-output

  --verbose-output = *`FILE`*
  Schreibt die ausführliche Ausgabe in *`FILE`*.

- version

  --version
  Zeigt die Version von RaptorXML Server an. Setzen Sie `--version` bei Verwendung mit einem Befehl vor den Befehl.

- warning-limit

  --warning-limit = *N | unlimited*
  Definiert das Warnungslimit im Bereich von `1-65535` oder `unbegrenzt`. Bei Erreichen dieses Limits wird die Verarbeitung fortgesetzt, doch werden keine weiteren Warnungen mehr ausgegeben. Der Standardwert ist `100`.

5.1.4 valxsd (xsd)

Der Befehl **valxsd | xsd** validiert ein oder mehrere XML-Schema-Dokumente (XSD-Dokumente) anhand der W3C XML Schema Definition Language (XSD) Spezifikationen 1.0 und 1.1. Beachten Sie, dass das Schema selbst anhand der XML-Schema-Spezifikation validiert wird und nicht ein XML-Instanzdokument.

```
raptorxml valxsd | xsd [options] InputFile
```

- Das Argument *InputFile* ist das zu validierende XML-Schema-Dokument. Die Option `--xsd-version=1.0|1.1|detect`[255] gibt an, anhand welcher XSD-Version das Dokument validiert werden soll, wobei der Standardwert `1.0` ist.
- Um mehrere Dokumente zu validieren, (i) listen Sie entweder die zu validierenden Dateien im CLI auf, wobei jede Datei durch ein Leerzeichen von der nächsten getrennt wird, oder (ii) listen Sie die zu validierenden Dateien in einer Textdatei (`.txt`-Datei) auf (ein Dateiname pro Zeile) und geben Sie diese Textdatei zusammen mit der auf `true` gesetzten Option `--listfile`[252] als das *InputFile* Argument an (*siehe Optionsliste unten*).

Beispiele

Beispiele für den Befehl **valxsd**:

- **raptorxml** valxsd c:\Test.xsd
- **raptorxml** xsd --verbose=true c:\Test.xsd
- **raptorxml** xsd --listfile=true c:\FileList.txt

- Groß- und Kleinschreibung und Schrägstriche in der Befehlszeile

  RaptorXML (und **RaptorXMLServer** für Administrator-Befehle) *unter Windows*
  raptorxml und **raptorxmlserver** für Administrator-Befehle) *unter Windows und Unix (Linux, Mac)*

  * Beachten Sie, dass klein geschriebene Befehle (`raptorxml` und `raptorxmlserver`) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (`RaptorXML`) nur unter Windows und Mac ausgeführt werden.

*Verwenden Sie auf Linux und Mac-Systemen Schrägstriche und auf Windows-Systemen umgekehrte Schrägstriche.

- Umgekehrte Schrägstriche, Leerzeichen und Sonderzeichen auf Windows-Systemen

  Auf Windows-Systemen: Verwenden Sie bei Vorhandensein von Leerzeichen oder Sonerzeichen in Strings (z.B. in Datei- oder Ordnernamen oder Firmen-, Personen- oder Produktnamen) Anführungszeichen: z.B., **`"Meine Datei"`**. Beachten Sie jedoch, dass ein von einem doppelten Anführungszeichen gefolgter umgekehrter Schrägstrich (z.B: **`"C:\Mein Verzeichnis\"`**) eventuell nicht korrekt gelesen wird, da der umgekehrte Schrägstrich auch den Beginn einer Escape-Sequenz markiert und die Escape-Sequenz **`\"`** für ein doppeltes Anführungszeichen steht. Wenn Sie diese Zeichensequenz mit einem Escape versehen wollen, verwenden Sie einen vorangestellten umgekehrten Schrägstrich, wie den folgenden: **`\\"`**. Einfacher ausgedrückt: Wenn Sie einen Dateipfad, der Leerzeichen oder einen umgekehrten Schrägstrich am Ende enthält, schreiben müssen, so schreiben Sie diesen folgendermaßen: **`"C:\Mein Verzeichnis\\"`**.

Optionen

Optionen werden in ihrer kurzen Form (falls verfügbar) und in ihrer langen Form aufgelistet. Für die kurze und die lange Form können ein oder zwei Bindestriche verwendet werden. Eine Option kann, muss aber keinen Wert erhalten. Eine Option, die einen Wert erhält, wird folgendermaßen geschrieben: **`--option=wert`**. Werte können außer in zwei Fällen ohne Anführungszeichen definiert werden: (i) wenn der Wertestring Leerzeichen enthält oder (ii) wenn in der Beschreibung der Option explizit erwähnt ist, dass Anführungszeichen zwingend erforderlich sind. Wenn eine Option einen Booleschen Wert erhält und kein Wert definiert ist, so ist der Standardwert der Option `TRUE`. Mit Hilfe der Option **`--h, --help`** können Sie Informationen über den Befehl anzeigen.

- Validierung und Verarbeitung
  - ct-restrict-mode

    `--ct-restrict-mode =` `1.0|1.1|default`
    Definiert, wie complexType-Einschränkungen überprüft werden sollen. Beim Wert **`1.0`** werden complexType-Einschränkungen anhand der XSD 1.0-Spezifikation überprüft - und zwar auch im XSD 1.1-Validierungsmodus. Beim Wert **`1.1`** werden complexType-Einschränkungen anhand der XSD 1.1-Spezifikation überprüft - und zwar auch im XSD 1.0-Validierungsmodus. Beim Wert **`default`** werden complexType-Einschränkungen anhand der als aktueller Validierungsmodus (1.0 oder 1.1) ausgewählten XSD-Spezifikation überprüft. Der Standardwert ist **`default`**.

  - listfile

    `--listfile =` `true|false`
    Bei `true` wird das Argument *`InputFile`* des Befehls als Textdatei behandelt, die einen Dateinamen pro Zeile enthält. Der Standardwert ist `false`. (Als Alternative können die Dateien im CLI getrennt durch ein Leerzeichen aufgelistet werden. Beachten Sie allerdings, dass CLIs eine maximale Zeichenanzahl haben). Beachten Sie, dass die Option `--listfile` nur auf Argumente, nicht aber auf Optionen angewendet wird.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

  - recurse

--recurse = true|false
Dient zur Auswahl von Dateien innerhalb von Unterverzeichnissen einschließlich ZIP-Archiven. Bei true wählt das Argument *InputFile* des Befehls die angegebene Datei auch in den Unterverzeichnissen aus. Beispiel: "test.zip|zip\test.xml" wählt Dateien mit dem Namen test.xml auf allen Ordnerebenen des ZIP-Ordners aus. Referenzen auf ZIP-Dateien müssen in Anführungszeichen angegeben werden. Es können die Platzhalter * und ? verwendet werden. Mit *.xml werden folglich alle .xml Dateien im (ZIP-)Ordner ausgewählt. Der Standardwert der Option ist false.
Hinweis: Die Booleschen Optionswerte werden auf true gesetzt, wenn die Option ohne einen Wert definiert wird.

- report-import-namespace-mismatch-as-warning

--report-import-namespace-mismatch-as-warning = true|false
Stuft Fehler, die beim Import von Schemas mit xs:import aufgrund eines nicht übereinstimmenden Namespace oder Ziel-Namespace auftreten, von Fehlern auf Warnungen herab. Der Standardwert ist false.
Hinweis: Die Booleschen Optionswerte werden auf true gesetzt, wenn die Option ohne einen Wert definiert wird.

- schema-imports

--schema-imports = load-by-schemalocation | load-preferring-schemalocation | load-by-namespace | load-combining-both | license-namespace-only
Definiert das Verhalten von xs:import Elementen, von denen jedes ein optionales namespace Attribut und ein optionales schemaLocation Attribut hat: <import namespace="someNS" schemaLocation="someURL">. Mit der Option wird definiert, ob ein Schema-Dokument geladen oder nur ein Namespace lizenziert werden soll und, wenn ein Schema-Dokument geladen werden soll, anhand welcher Informationen es gesucht werden soll. Standardeinstellung: load-preferring-schemalocation. Das Verhalten ist das folgende:

- load-by-schemalocation: Anhand des Werts des schemaLocation Attributs wird der Schemapfad ermittelt, wobei Katalog-Mappings[48] berücksichtigt werden. Wenn das Namespace-Attribut vorhanden ist, wird der Namespace importiert (lizenziert).
- load-preferring-schemalocation: Wenn das schemaLocation Attribut vorhanden ist, wird es verwendet, wobei Katalog-Mappings[48] berücksichtigt werden. Falls kein schemaLocation Attribut vorhanden ist, wird der Wert des namespace Attributs über ein Katalog-Mapping[48] verwendet. Dies ist der **Standardwert**.
- load-by-namespace: Anhand des Werts des namespace Attributs wird der Schemapfad über ein Katalog-Mapping[48] ermittelt.
- load-combining-both: Wenn entweder das Attribut namespace oder das Attribut schemaLocation ein Katalog-Mapping[48] hat, so wird das Mapping verwendet. Wenn beide Attribute Katalog-Mappings[48] haben, ist es vom Wert der Option --schema-mapping (XML/XSD-Option[255]) abhängig, welches Mapping verwendet wird. Falls kein Katalog-Mapping[48] vorhanden ist, wird das schemaLocation Attribut verwendet.
- license-namespace-only: Der Namespace wird importiert. Kein Schema-Dokument wird importiert.

- schema-location-hints

--schema-location-hints = load-by-schemalocation | load-by-namespace | load-combining-both | ignore

Definiert das Verhalten der Attribute `xsi:schemaLocation` und `xsi:noNamespaceSchemaLocation`: Legt fest, ob ein Schema-Dokument geladen werden soll und falls ja, anhand welcher Informationen es gesucht werden soll. Standardeinstellung: `load-by-schemalocation`.

- Der Wert `load-by-schemalocation` verwendet die URL des Schemapfads[422] in den Attributen `xsi:schemaLocation` und `xsi:noNamespaceSchemaLocation` in XML-Instanzdokumenten. Dies ist der **Standardwert**.
- Der Wert `load-by-namespace` verwendet den Namespace-Teil[422] von `xsi:schemaLocation` und - im Fall von `xsi:noNamespaceSchemaLocation` - einen leeren String und ermittelt das Schema über ein Katalog-Mapping[48].
- Bei Verwendung von `load-combining-both` und wenn entweder der Namespace-Teil oder der URL-Teil ein Katalog-Mapping[48] hat, so wird das Katalog-Mapping[48] verwendet. Wenn beide Katalog-Mappings[48] haben, ist es vom Wert der `--schema-mapping` Option (XML/XSD-Option[255]) abhängig, welches Mapping verwendet wird. Wenn weder der Namespace noch die URL ein Katalog-Mapping hat, wird die URL verwendet.
- Wenn der Wert der Option `ignore` ist, werden die beiden Attribute `xsi:schemaLocation` und `xsi:noNamespaceSchemaLocation` ignoriert.

▼ schema-mapping

`--schema-mapping =` `prefer-schemalocation | prefer-namespace`
Wenn ein Schema-Dokument sowohl anhand des Schema-Pfads als auch des Namespace gesucht werden soll, wird damit festgelegt, welche der beiden Optionen beim Katalog-Lookup Vorrang erhält. (Wenn eine der Optionen `--schemalocation-hints` oder `--schema-imports` einen Wert `load-combining-both` hat und wenn die betroffenen Namespace- und URL-Teile beide Katalog-Mappings[48] haben, gibt der Wert dieser Option an, welches der beiden Mappings verwendet werden soll (das Namespace Mapping oder das URL-Mapping; der Wert `prefer-schemalocation` bezieht sich auf das URL-Mapping)). Der Standardwert ist `prefer-schemalocation`.

▼ script

`--script =` *`FILE`*
Führt nach Abschluss der Validierung das Python-Skript in der angegebenen Datei aus. Fügen Sie die Option mehrmals hinzu, um mehr als ein Skript zu definieren.

▼ script-api-version

`--api, --script-api-version =` `1; 2; 2.1 to 2.4; 2.4.1; 2.5 bis 2.8; 2.8.1 bis 2.8.6; 2.9.0; 2.10.0; 2.11.0`
Definiert, welche Python API-Version für das Skript verwendet werden soll. Der Standardwert ist die neueste Version, derzeit **`2.11.0`**. Anstelle von Ganzzahlwerten wie `1` und `2` können Sie auch die entsprechenden Werte `1.0` und `2.0` verwenden. Ebenso können Sie anstelle der zwei Ziffern `2.5` die drei Ziffern `2.5.0` verwenden. Siehe auch Kapitel Python API-Versionen[392].

▼ script-output

`--script-output =` *`FILE`*
Schreibt die Standardausgabe des Skripts in die in *`FILE`* angegebene Datei.

▼ script-param

`--script-param =` *`KEY:VALUE`*
Zusätzliche benutzerdefinierte Parameter, die während der Ausführung von Python Skripts aufgerufen werden können. Fügen Sie die Option mehrmals hinzu, um mehr als einen Parameter zu definieren.

- xinclude

  --xinclude = `true|false`
  Aktiviert die Unterstützung für XML-Inkludierungen (XInclude). Der Standardwert ist `false`. Bei `false` werden die `include`-Elemente von XInclude ignoriert.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- xml-mode-for-schemas

  --xml-mode-for-schemas = `wf|id|valid`
  Definiert, welcher XML-Verarbeitungsmodus für XML-Schema-Dokumente verwendet werden soll: `wf`=Wohlgeformtheitsprüfung; `id`=Wohlgeformtheitsprüfung mit ID/IDREF-Prüfung; `valid`=Validierung. Der Standardwert ist `wf`. Beachten Sie, dass jedes bei der Verarbeitung geladene Schema-Dokument eine DTD referenzieren muss, damit der Wert `valid` ausgegeben werden kann. Falls keine DTD vorhanden ist, wird ein Fehler ausgegeben.

- xsd-version

  --xsd-version = `1.0|1.1|detect`
  Definiert die zu verwendende W3C Schema Definition Language (XSD) Version. Der Standardwert ist `1.0`. Diese Option eignet sich auch, um herauszufinden, inwiefern ein 1.0-kompatibles Schema nicht mit Schemaversion 1.1 kompatibel ist. Die Option `detect` ist eine Altova-spezifische Funktionalität. Mit dieser Option kann die Version des XML-Schema-Dokuments (`1.0` oder `1.1`) durch Lesen des Werts des `vc:minVersion` Attributs des `<xs:schema>` Elements des Dokuments ermittelt werden. Wenn der Wert des `@vc:minVersion` Attributs `1.1` ist, wird das Schema als Version `1.1` erkannt. Bei jedem anderen Wert wird das Schema als `1.0` erkannt. Bei jedem anderen Wert oder bei Fehlen des `@vc:minVersion` Attributs wird das Schema als Version `1.0` gelesen.

- Kataloge und globale Ressourcen
  - catalog

    --catalog = *FILE*
    Gibt den absoluten Pfad zu einer Root-Katalog-Datei an, die nicht die installierte Root-Katalog-Datei ist. Der Standardwert ist der absolute Pfad zur installierten Root-Katalog-Datei. (`<installationsordner>\Altova\RaptorXMLServer2025\etc\RootCatalog.xml`). Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge 48.

  - user-catalog

    --user-catalog = *FILE*
    Definiert den absoluten Pfad zu einem XML-Katalog, der zusätzlich zum Root-Katalog verwendet werden soll. Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge 48.

  - enable-globalresources

    --enable-globalresources = `true|false`
    Aktiviert die globalen Ressourcen 55. Standardwert ist `false`.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- globalresourceconfig [gc]

  --gc | --globalresourceconfig = *VALUE*
  Definiert die aktive Konfiguration der globalen Ressource[55] (und aktiviert globale Ressourcen)[55].

- globalresourcefile [gr]

  --gr | --globalresourcefile = *FILE*
  Definiert die globale Ressourcendatei[55] (und aktiviert globale Ressourcen)[55].

- Allgemeine Optionen
  - error-format

    --error-format = `text|shortxml|longxml`
    Definiert das Format der Fehlerausgabe. Der Standardwert ist `text`. Mit den anderen Optionen werden XML-Formate generiert, wobei mit `longxml` mehr Details generiert werden.

  - error-limit

    --error-limit = *N | unlimited*
    Definiert das Fehlerlimit mit einem Wertebereich von `1` bis `9999` oder `unbegrenzt`. Der Standardwert ist `100`. Bei Erreichung des Fehlerlimits wird die Validierung gestoppt. Dient dazu, die Prozessorverwendung während der Validierung/Transformation einzuschränken.

  - info-limit

    --info-limit = *N | unlimited*
    Definiert die Grenze für Informationsmeldungen im Bereich von `1-65535` oder als `unlimited` (unbegrenzt). Die Verarbeitung wird auch nach Erreichen des Info-Limits fortgesetzt, doch werden weitere Meldungen nicht mehr ausgegeben. Der Standardwert ist `100`.

  - help

    --help
    Zeigt den Hilfetext zum Befehl an. Beispiel: `valany --h`. (Alternativ dazu kann der Befehl `help` zusammen mit einem Argument verwendet werden. Beispiel: `help valany`.)

  - listfile

    --listfile = `true|false`
    Bei `true` wird das Argument *InputFile* des Befehls als Textdatei behandelt, die einen Dateinamen pro Zeile enthält. Der Standardwert ist `false`. (Als Alternative können die Dateien im CLI getrennt durch ein Leerzeichen aufgelistet werden. Beachten Sie allerdings, dass CLIs eine maximale Zeichenanzahl haben). Beachten Sie, dass die Option `--listfile` nur auf Argumente, nicht aber auf Optionen angewendet wird.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

  - log-output

    --log-output = *FILE*
    Schreibt die Meldungsausgabe in die angegebene URL. Stellen Sie sicher, dass das CLI

Schreibrechte für den Ausgabepfad hat.

- network-timeout

  --network-timeout = *VALUE*
  Definiert das Timeout für entfernte I/O-Operationen in Millisekunden. Der Standardwert ist: `40000`.

- recurse

  --recurse = `true|false`
  Dient zur Auswahl von Dateien innerhalb von Unterverzeichnissen einschließlich ZIP-Archiven. Bei `true` wählt das Argument *InputFile* des Befehls die angegebene Datei auch in den Unterverzeichnissen aus. Beispiel: `"test.zip|zip\test.xml"` wählt Dateien mit dem Namen `test.xml` auf allen Ordnerebenen des ZIP-Ordners aus. Referenzen auf ZIP-Dateien müssen in Anführungszeichen angegeben werden. Es können die Platzhalter `*` und `?` verwendet werden. Mit `*.xml` werden folglich alle `.xml` Dateien im (ZIP-)Ordner ausgewählt. Der Standardwert der Option ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- verbose

  --verbose = `true|false`
  Mit dem Wert `true` wird die Ausgabe zusätzlicher Informationen bei der Validierung aktiviert. Der Standardwert ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- verbose-output

  --verbose-output = *FILE*
  Schreibt die ausführliche Ausgabe in *FILE*.

- version

  --version
  Zeigt die Version von RaptorXML Server an. Setzen Sie `--version` bei Verwendung mit einem Befehl vor den Befehl.

- warning-limit

  --warning-limit = *N | unlimited*
  Definiert das Warnungslimit im Bereich von `1-65535` oder `unbegrenzt`. Bei Erreichen dieses Limits wird die Verarbeitung fortgesetzt, doch werden keine weiteren Warnungen mehr ausgegeben. Der Standardwert ist `100`.

5.2 Befehle für die Überprüfung der Wohlgeformtheit

Mit Hilfe der Befehle zur Überprüfung der Wohlgeformtheit kann die Wohlgeformtheit von XML-Dokumenten und DTDs überprüft werden. Diese Befehle sind unten aufgelistet und in den Unterabschnitten dieses Abschnitts näher beschrieben:

- wfxml 84: Überprüft ein XML-Dokument auf Wohlgeformtheit.
- wfdtd 89: Überprüft ein DTD-Dokument auf Wohlgeformtheit.
- wfany 93: Überprüft jedes XML- oder DTD-Dokument auf Wohlgeformtheit. Der Typ des Dokuments wird automatisch erkannt.

5.2.1 wfxml

Der Befehl **wfxml** überprüft ein oder mehrere XML-Dokumente anhand der XML 1.0- oder XML 1.1-Spezifikation auf Wohlgeformtheit.

```
raptorxml wfxml [options] InputFile
```

- Das Argument *InputFile* ist das XML-Dokument, das auf Wohlgeformtheit überprüft werden soll.
- Um mehrere Input-Dokumente auf Wohlgeformtheit zu prüfen, (i) listen Sie entweder die zu überprüfenden Dateien im CLI auf, wobei jede Datei durch ein Leerzeichen von der nächsten getrennt wird, oder (ii) listen Sie die zu überprüfenden Dateien in einer Textdatei (`.txt`-Datei) auf (ein Dateiname pro Zeile) und geben Sie diese Textdatei zusammen mit der auf `true` gesetzten Option `--listfile` 252 als das *InputFile* Argument an (*siehe Optionsliste unten*).

Beispiele

Beispiele für den Befehl **wfxml**:

- `raptorxml wfxml c:\Test.xml`
- **raptorxml** `wfxml --verbose=true c:\Test.xml`
- **raptorxml** `wfxml --listfile=true c:\FileList.txt`

▼ Groß- und Kleinschreibung und Schrägstriche in der Befehlszeile

RaptorXML (und **RaptorXMLServer** für Administrator-Befehle) *unter Windows*
raptorxml und **raptorxmlserver** für Administrator-Befehle) *unter Windows und Unix (Linux, Mac)*

* Beachten Sie, dass klein geschriebene Befehle (`raptorxml` und `raptorxmlserver`) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (`RaptorXML`) nur unter Windows und Mac ausgeführt werden.
*Verwenden Sie auf Linux und Mac-Systemen Schrägstriche und auf Windows-Systemen umgekehrte Schrägstriche.

▼ Umgekehrte Schrägstriche, Leerzeichen und Sonderzeichen auf Windows-Systemen

Auf Windows-Systemen: Verwenden Sie bei Vorhandensein von Leerzeichen oder Sonerzeichen in Strings (z.B. in Datei- oder Ordnernamen oder Firmen-, Personen- oder Produktnamen) Anführungszeichen: z.B., **"Meine Datei"**. Beachten Sie jedoch, dass ein von einem doppelten Anführungszeichen gefolgter umgekehrter Schrägstrich (z.B: **"C:\Mein Verzeichnis\"**) eventuell nicht

korrekt gelesen wird, da der umgekehrte Schrägstrich auch den Beginn einer Escape-Sequenz markiert und die Escape-Sequenz **\"** für ein doppeltes Anführungszeichen steht. Wenn Sie diese Zeichensequenz mit einem Escape versehen wollen, verwenden Sie einen vorangestellten umgekehrten Schrägstrich, wie den folgenden: **\\"**. Einfacher ausgedrückt: Wenn Sie einen Dateipfad, der Leerzeichen oder einen umgekehrten Schrägstrich am Ende enthält, schreiben müssen, so schreiben Sie diesen folgendermaßen: **"C:\Mein Verzeichnis\\"**.

Optionen

Optionen werden in ihrer kurzen Form (falls verfügbar) und in ihrer langen Form aufgelistet. Für die kurze und die lange Form können ein oder zwei Bindestriche verwendet werden. Eine Option kann, muss aber keinen Wert erhalten. Eine Option, die einen Wert erhält, wird folgendermaßen geschrieben: **--option=wert**. Werte können außer in zwei Fällen ohne Anführungszeichen definiert werden: (i) wenn der Wertestring Leerzeichen enthält oder (ii) wenn in der Beschreibung der Option explizit erwähnt ist, dass Anführungszeichen zwingend erforderlich sind. Wenn eine Option einen Booleschen Wert erhält und kein Wert definiert ist, so ist der Standardwert der Option `TRUE`. Mit Hilfe der Option **--h, --help** können Sie Informationen über den Befehl anzeigen.

- Validierung und Verarbeitung
  - dtd

    --dtd = *FILE*
    Definiert das für die Validierung zu verwendende externe DTD-Dokument. Wenn das XML-Dokument eine Referenz auf eine externe DTD enthält, setzt die CLI-Option die externe Referenz außer Kraft.

  - listfile

    --listfile = `true|false`
    Bei `true` wird das Argument *InputFile* des Befehls als Textdatei behandelt, die einen Dateinamen pro Zeile enthält. Der Standardwert ist `false`. (Als Alternative können die Dateien im CLI getrennt durch ein Leerzeichen aufgelistet werden. Beachten Sie allerdings, dass CLIs eine maximale Zeichenanzahl haben). Beachten Sie, dass die Option `--listfile` nur auf Argumente, nicht aber auf Optionen angewendet wird.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

  - namespaces

    --namespaces = `true|false`
    Aktiviert die Verarbeitung unter Berücksichtigung des Namespace. Dies ist nützlich, um die XML-Instanz auf Fehler aufgrund falscher Namespaces zu überprüfen. Der Standardwert ist `false`.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

  - recurse

    --recurse = `true|false`
    Dient zur Auswahl von Dateien innerhalb von Unterverzeichnissen einschließlich ZIP-Archiven. Bei `true` wählt das Argument *InputFile* des Befehls die angegebene Datei auch in den

Unterverzeichnissen aus. Beispiel: `"test.zip|zip\test.xml"` wählt Dateien mit dem Namen `test.xml` auf allen Ordnerebenen des ZIP-Ordners aus. Referenzen auf ZIP-Dateien müssen in Anführungszeichen angegeben werden. Es können die Platzhalter `*` und `?` verwendet werden. Mit `*.xml` werden folglich alle `.xml` Dateien im (ZIP-)Ordner ausgewählt. Der Standardwert der Option ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- script

  `--script =` *FILE*
  Führt nach Abschluss der Validierung das Python-Skript in der angegebenen Datei aus. Fügen Sie die Option mehrmals hinzu, um mehr als ein Skript zu definieren.

- script-api-version

  `--api, --script-api-version =` `1; 2; 2.1 to 2.4; 2.4.1; 2.5 bis 2.8; 2.8.1 bis 2.8.6; 2.9.0; 2.10.0; 2.11.0`
  Definiert, welche Python API-Version für das Skript verwendet werden soll. Der Standardwert ist die neueste Version, derzeit **`2.11.0`**. Anstelle von Ganzzahlwerten wie `1` und `2` können Sie auch die entsprechenden Werte `1.0` und `2.0` verwenden. Ebenso können Sie anstelle der zwei Ziffern `2.5` die drei Ziffern `2.5.0` verwenden. Siehe auch Kapitel Python API-Versionen 392.

- script-output

  `--script-output =` *FILE*
  Schreibt die Standardausgabe des Skripts in die in *FILE* angegebene Datei.

- script-param

  `--script-param =` *KEY:VALUE*
  Zusätzliche benutzerdefinierte Parameter, die während der Ausführung von Python Skripts aufgerufen werden können. Fügen Sie die Option mehrmals hinzu, um mehr als einen Parameter zu definieren.

- streaming

  `--streaming =` `true|false`
  Aktiviert die Streaming-Validierung. Standardwert ist `true`. Die im Arbeitsspeicher gehaltene Datenmenge wird im Streaming-Modus minimiert. Der Nachteil ist, dass später eventuell benötigte Informationen - z.B. ein Datenmodell des XML-Instanzdokuments - nicht mehr verfügbar sind. In Situationen, in denen dies eine Rolle spielt, muss der Streaming-Modus deaktiviert werden (indem Sie `--streaming` auf den Wert `false` setzen). Wenn Sie die Option `--script` mit dem Befehl `valxml-withxsd` verwenden, sollten Sie das Streaming deaktivieren. Beachten Sie, dass die Option `--streaming` ignoriert wird, wenn `--parallel-assessment` auf `true` gesetzt wird.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

Kataloge und globale Ressourcen

- catalog

  `--catalog =` *FILE*

Gibt den absoluten Pfad zu einer Root-Katalog-Datei an, die nicht die installierte Root-Katalog-Datei ist. Der Standardwert ist der absolute Pfad zur installierten Root-Katalog-Datei. (`<installationsordner>\Altova\RaptorXMLServer2025\etc\RootCatalog.xml`). Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge[48].

- user-catalog

  `--user-catalog =` *`FILE`*
  Definiert den absoluten Pfad zu einem XML-Katalog, der zusätzlich zum Root-Katalog verwendet werden soll. Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge[48].

- enable-globalresources

  `--enable-globalresources =` `true|false`
  Aktiviert die globalen Ressourcen[55]. Standardwert ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- globalresourceconfig [gc]

  `--gc | --globalresourceconfig =` *`VALUE`*
  Definiert die aktive Konfiguration der globalen Ressource[55] (und aktiviert globale Ressourcen)[55].

- globalresourcefile [gr]

  `--gr | --globalresourcefile =` *`FILE`*
  Definiert die globale Ressourcendatei[55] (und aktiviert globale Ressourcen)[55].

Allgemeine Optionen

- error-format

  `--error-format =` `text|shortxml|longxml`
  Definiert das Format der Fehlerausgabe. Der Standardwert ist `text`. Mit den anderen Optionen werden XML-Formate generiert, wobei mit `longxml` mehr Details generiert werden.

- error-limit

  `--error-limit =` *`N | unlimited`*
  Definiert das Fehlerlimit mit einem Wertebereich von `1` bis `9999` oder `unbegrenzt`. Der Standardwert ist `100`. Bei Erreichung des Fehlerlimits wird die Validierung gestoppt. Dient dazu, die Prozessorverwendung während der Validierung/Transformation einzuschränken.

- info-limit

  `--info-limit =` *`N | unlimited`*
  Definiert die Grenze für Informationsmeldungen im Bereich von `1-65535` oder als `unlimited` (unbegrenzt). Die Verarbeitung wird auch nach Erreichen des Info-Limits fortgesetzt, doch werden weitere Meldungen nicht mehr ausgegeben. Der Standardwert ist `100`.

- help

--help
Zeigt den Hilfetext zum Befehl an. Beispiel: `valany --h`. (Alternativ dazu kann der Befehl `help` zusammen mit einem Argument verwendet werden. Beispiel: `help valany`.)

- listfile

  --listfile = `true|false`
  Bei `true` wird das Argument *`InputFile`* des Befehls als Textdatei behandelt, die einen Dateinamen pro Zeile enthält. Der Standardwert ist `false`. (Als Alternative können die Dateien im CLI getrennt durch ein Leerzeichen aufgelistet werden. Beachten Sie allerdings, dass CLIs eine maximale Zeichenanzahl haben). Beachten Sie, dass die Option `--listfile` nur auf Argumente, nicht aber auf Optionen angewendet wird.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- log-output

  --log-output = *`FILE`*
  Schreibt die Meldungsausgabe in die angegebene URL. Stellen Sie sicher, dass das CLI Schreibrechte für den Ausgabepfad hat.

- network-timeout

  --network-timeout = *`VALUE`*
  Definiert das Timeout für entfernte I/O-Operationen in Millisekunden. Der Standardwert ist: `40000`.

- recurse

  --recurse = `true|false`
  Dient zur Auswahl von Dateien innerhalb von Unterverzeichnissen einschließlich ZIP-Archiven. Bei `true` wählt das Argument *`InputFile`* des Befehls die angegebene Datei auch in den Unterverzeichnissen aus. Beispiel: `"test.zip|zip\test.xml"` wählt Dateien mit dem Namen `test.xml` auf allen Ordnerebenen des ZIP-Ordners aus. Referenzen auf ZIP-Dateien müssen in Anführungszeichen angegeben werden. Es können die Platzhalter `*` und `?` verwendet werden. Mit `*.xml` werden folglich alle `.xml` Dateien im (ZIP-)Ordner ausgewählt. Der Standardwert der Option ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- verbose

  --verbose = `true|false`
  Mit dem Wert `true` wird die Ausgabe zusätzlicher Informationen bei der Validierung aktiviert. Der Standardwert ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- verbose-output

  --verbose-output = *`FILE`*
  Schreibt die ausführliche Ausgabe in *`FILE`*.

- version

--version
Zeigt die Version von RaptorXML Server an. Setzen Sie `--version` bei Verwendung mit einem Befehl vor den Befehl.

- warning-limit

  --warning-limit = *N | unlimited*
  Definiert das Warnungslimit im Bereich von `1-65535` oder `unbegrenzt`. Bei Erreichen dieses Limits wird die Verarbeitung fortgesetzt, doch werden keine weiteren Warnungen mehr ausgegeben. Der Standardwert ist `100`.

5.2.2 wfdtd

Der Befehl **wfdtd** überprüft ein oder mehrere DTD-Dokumente anhand der XML 1.0- oder XML 1.1-Spezifikation auf Wohlgeformtheit.

```
raptorxml wfdtd [options] InputFile
```

- Das Argument *InputFile* ist das zu überprüfende DTD-Dokument.
- Um mehrere Dokumente auf Wohlgeformtheit zu prüfen, (i) listen Sie entweder die zu überprüfenden Dateien im CLI auf, wobei jede Datei durch ein Leerzeichen von der nächsten getrennt wird, oder (ii) listen Sie die zu überprüfenden Dateien in einer Textdatei (`.txt`-Datei) auf (ein Dateiname pro Zeile) und geben Sie diese Textdatei zusammen mit der auf `true` gesetzten Option `--listfile` 252 als das *InputFile* Argument an (*siehe Optionsliste unten*).

Beispiele

Beispiele für den Befehl **wfdtd**:

- **raptorxml** wfdtd c:\Test.dtd
- **raptorxml** wfdtd --verbose=true c:\Test.dtd
- **raptorxml** wfdtd --listfile=true c:\FileList.txt

- Groß- und Kleinschreibung und Schrägstriche in der Befehlszeile

  RaptorXML (und **RaptorXMLServer** für Administrator-Befehle) *unter Windows*
  raptorxml und **raptorxmlserver** für Administrator-Befehle) *unter Windows und Unix (Linux, Mac)*

  * Beachten Sie, dass klein geschriebene Befehle (`raptorxml` und `raptorxmlserver`) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (`RaptorXML`) nur unter Windows und Mac ausgeführt werden.
  *Verwenden Sie auf Linux und Mac-Systemen Schrägstriche und auf Windows-Systemen umgekehrte Schrägstriche.

- Umgekehrte Schrägstriche, Leerzeichen und Sonderzeichen auf Windows-Systemen

  Auf Windows-Systemen: Verwenden Sie bei Vorhandensein von Leerzeichen oder Sonerzeichen in Strings (z.B. in Datei- oder Ordnernamen oder Firmen-, Personen- oder Produktnamen) Anführungszeichen: z.B., **`"Meine Datei"`**. Beachten Sie jedoch, dass ein von einem doppelten Anführungszeichen gefolgter umgekehrter Schrägstrich (z.B: **`"C:\Mein Verzeichnis\"`**) eventuell nicht korrekt gelesen wird, da der umgekehrte Schrägstrich auch den Beginn einer Escape-Sequenz markiert und die Escape-Sequenz **`\"`** für ein doppeltes Anführungszeichen steht. Wenn Sie diese Zeichensequenz mit einem Escape versehen wollen, verwenden Sie einen vorangestellten umgekehrten Schrägstrich, wie den folgenden: **`\\"`**. Einfacher ausgedrückt: Wenn Sie einen Dateipfad, der Leerzeichen oder einen umgekehrten Schrägstrich am Ende enthält, schreiben müssen, so schreiben Sie diesen folgendermaßen: **`"C:\Mein Verzeichnis\\"`**.

Optionen

Optionen werden in ihrer kurzen Form (falls verfügbar) und in ihrer langen Form aufgelistet. Für die kurze und die lange Form können ein oder zwei Bindestriche verwendet werden. Eine Option kann, muss aber keinen Wert erhalten. Eine Option, die einen Wert erhält, wird folgendermaßen geschrieben: **`--option=wert`**. Werte können außer in zwei Fällen ohne Anführungszeichen definiert werden: (i) wenn der Wertestring Leerzeichen enthält oder (ii) wenn in der Beschreibung der Option explizit erwähnt ist, dass Anführungszeichen zwingend erforderlich sind. Wenn eine Option einen Booleschen Wert erhält und kein Wert definiert ist, so ist der Standardwert der Option `TRUE`. Mit Hilfe der Option **`--h, --help`** können Sie Informationen über den Befehl anzeigen.

- Validierung und Verarbeitung
  - listfile

    `--listfile =` `true|false`
    Bei `true` wird das Argument *`InputFile`* des Befehls als Textdatei behandelt, die einen Dateinamen pro Zeile enthält. Der Standardwert ist `false`. (Als Alternative können die Dateien im CLI getrennt durch ein Leerzeichen aufgelistet werden. Beachten Sie allerdings, dass CLIs eine maximale Zeichenanzahl haben). Beachten Sie, dass die Option `--listfile` nur auf Argumente, nicht aber auf Optionen angewendet wird.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

  - recurse

    `--recurse =` `true|false`
    Dient zur Auswahl von Dateien innerhalb von Unterverzeichnissen einschließlich ZIP-Archiven. Bei `true` wählt das Argument *`InputFile`* des Befehls die angegebene Datei auch in den Unterverzeichnissen aus. Beispiel: `"test.zip|zip\test.xml"` wählt Dateien mit dem Namen `test.xml` auf allen Ordnerebenen des ZIP-Ordners aus. Referenzen auf ZIP-Dateien müssen in Anführungszeichen angegeben werden. Es können die Platzhalter `*` und `?` verwendet werden. Mit `*.xml` werden folglich alle `.xml` Dateien im (ZIP-)Ordner ausgewählt. Der Standardwert der Option ist `false`.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- Kataloge und globale Ressourcen
  - catalog

    --catalog = *FILE*
    Gibt den absoluten Pfad zu einer Root-Katalog-Datei an, die nicht die installierte Root-Katalog-Datei ist. Der Standardwert ist der absolute Pfad zur installierten Root-Katalog-Datei. (`<installationsordner>\Altova\RaptorXMLServer2025\etc\RootCatalog.xml`). Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge[48].

  - user-catalog

    --user-catalog = *FILE*
    Definiert den absoluten Pfad zu einem XML-Katalog, der zusätzlich zum Root-Katalog verwendet werden soll. Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge[48].

  - enable-globalresources

    --enable-globalresources = `true|false`
    Aktiviert die globalen Ressourcen[55]. Standardwert ist `false`.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

  - globalresourceconfig [gc]

    --gc | --globalresourceconfig = *VALUE*
    Definiert die aktive Konfiguration der globalen Ressource[55] (und aktiviert globale Ressourcen)[55].

  - globalresourcefile [gr]

    --gr | --globalresourcefile = *FILE*
    Definiert die globale Ressourcendatei[55] (und aktiviert globale Ressourcen)[55].

- Allgemeine Optionen
  - error-format

    --error-format = `text|shortxml|longxml`
    Definiert das Format der Fehlerausgabe. Der Standardwert ist `text`. Mit den anderen Optionen werden XML-Formate generiert, wobei mit `longxml` mehr Details generiert werden.

  - error-limit

    --error-limit = *N | unlimited*
    Definiert das Fehlerlimit mit einem Wertebereich von `1` bis `9999` oder `unbegrenzt`. Der Standardwert ist `100`. Bei Erreichung des Fehlerlimits wird die Validierung gestoppt. Dient dazu, die Prozessorverwendung während der Validierung/Transformation einzuschränken.

  - info-limit

    --info-limit = *N | unlimited*
    Definiert die Grenze für Informationsmeldungen im Bereich von `1-65535` oder als `unlimited` (unbegrenzt). Die Verarbeitung wird auch nach Erreichen des Info-Limits fortgesetzt, doch werden weitere Meldungen nicht mehr ausgegeben. Der Standardwert ist `100`.

- help

  --help
  Zeigt den Hilfetext zum Befehl an. Beispiel: `valany --h`. (Alternativ dazu kann der Befehl `help` zusammen mit einem Argument verwendet werden. Beispiel: `help valany`.)

- listfile

  --listfile = `true|false`
  Bei `true` wird das Argument *InputFile* des Befehls als Textdatei behandelt, die einen Dateinamen pro Zeile enthält. Der Standardwert ist `false`. (Als Alternative können die Dateien im CLI getrennt durch ein Leerzeichen aufgelistet werden. Beachten Sie allerdings, dass CLIs eine maximale Zeichenanzahl haben). Beachten Sie, dass die Option `--listfile` nur auf Argumente, nicht aber auf Optionen angewendet wird.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- log-output

  --log-output = *FILE*
  Schreibt die Meldungsausgabe in die angegebene URL. Stellen Sie sicher, dass das CLI Schreibrechte für den Ausgabepfad hat.

- network-timeout

  --network-timeout = *VALUE*
  Definiert das Timeout für entfernte I/O-Operationen in Millisekunden. Der Standardwert ist: `40000`.

- recurse

  --recurse = `true|false`
  Dient zur Auswahl von Dateien innerhalb von Unterverzeichnissen einschließlich ZIP-Archiven. Bei `true` wählt das Argument *InputFile* des Befehls die angegebene Datei auch in den Unterverzeichnissen aus. Beispiel: `"test.zip|zip\test.xml"` wählt Dateien mit dem Namen `test.xml` auf allen Ordnerebenen des ZIP-Ordners aus. Referenzen auf ZIP-Dateien müssen in Anführungszeichen angegeben werden. Es können die Platzhalter `*` und `?` verwendet werden. Mit `*.xml` werden folglich alle `.xml` Dateien im (ZIP-)Ordner ausgewählt. Der Standardwert der Option ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- verbose

  --verbose = `true|false`
  Mit dem Wert `true` wird die Ausgabe zusätzlicher Informationen bei der Validierung aktiviert. Der Standardwert ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- verbose-output

  --verbose-output = *FILE*

Schreibt die ausführliche Ausgabe in *FILE*.

- version

  --version
  Zeigt die Version von RaptorXML Server an. Setzen Sie `--version` bei Verwendung mit einem Befehl vor den Befehl.

- warning-limit

  --warning-limit = *N | unlimited*
  Definiert das Warnungslimit im Bereich von `1-65535` oder `unbegrenzt`. Bei Erreichen dieses Limits wird die Verarbeitung fortgesetzt, doch werden keine weiteren Warnungen mehr ausgegeben. Der Standardwert ist `100`.

5.2.3 wfany

Der Befehl **wfany** überprüft die Wohlgeformtheit eines XML-, DTD- oder XML-Schema-Dokuments anhand der jeweiligen Spezifikation(en). Der Typ des Dokuments wird automatisch erkannt.

```
raptorxml wfany [options] InputFile
```

- Das Argument *InputFile* gibt das Dokument an, dessen Wohlgeformtheit geprüft werden soll.
- Beachten Sie, dass nur ein einziges Dokument als Argument des Befehls angegeben werden kann. Der Typ des angegebenen Dokuments wird automatisch erkannt.

Beispiele

Beispiele für den Befehl **wfany**:

- **raptorxml** wfany c:\Test.xml
- **raptorxml** wfany --error-format=text c:\Test.xml

- Groß- und Kleinschreibung und Schrägstriche in der Befehlszeile

  RaptorXML (und **RaptorXMLServer** für Administrator-Befehle) *unter Windows*
  raptorxml und **raptorxmlserver** für Administrator-Befehle) *unter Windows und Unix (Linux, Mac)*

  * Beachten Sie, dass klein geschriebene Befehle (`raptorxml` und `raptorxmlserver`) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (`RaptorXML`) nur unter Windows und Mac ausgeführt werden.
  *Verwenden Sie auf Linux und Mac-Systemen Schrägstriche und auf Windows-Systemen umgekehrte Schrägstriche.

- Umgekehrte Schrägstriche, Leerzeichen und Sonderzeichen auf Windows-Systemen

  Auf Windows-Systemen: Verwenden Sie bei Vorhandensein von Leerzeichen oder Sonerzeichen in Strings (z.B. in Datei- oder Ordnernamen oder Firmen-, Personen- oder Produktnamen)

Anführungszeichen: z.B., **"Meine Datei"**. Beachten Sie jedoch, dass ein von einem doppelten Anführungszeichen gefolgter umgekehrter Schrägstrich (z.B: **"C:\Mein Verzeichnis\"**) eventuell nicht korrekt gelesen wird, da der umgekehrte Schrägstrich auch den Beginn einer Escape-Sequenz markiert und die Escape-Sequenz **\"** für ein doppeltes Anführungszeichen steht. Wenn Sie diese Zeichensequenz mit einem Escape versehen wollen, verwenden Sie einen vorangestellten umgekehrten Schrägstrich, wie den folgenden: **\\"**. Einfacher ausgedrückt: Wenn Sie einen Dateipfad, der Leerzeichen oder einen umgekehrten Schrägstrich am Ende enthält, schreiben müssen, so schreiben Sie diesen folgendermaßen: **"C:\Mein Verzeichnis\\"**.

Optionen

Optionen werden in ihrer kurzen Form (falls verfügbar) und in ihrer langen Form aufgelistet. Für die kurze und die lange Form können ein oder zwei Bindestriche verwendet werden. Eine Option kann, muss aber keinen Wert erhalten. Eine Option, die einen Wert erhält, wird folgendermaßen geschrieben: **--option=wert**. Werte können außer in zwei Fällen ohne Anführungszeichen definiert werden: (i) wenn der Wertestring Leerzeichen enthält oder (ii) wenn in der Beschreibung der Option explizit erwähnt ist, dass Anführungszeichen zwingend erforderlich sind. Wenn eine Option einen Booleschen Wert erhält und kein Wert definiert ist, so ist der Standardwert der Option `TRUE`. Mit Hilfe der Option **--h, --help** können Sie Informationen über den Befehl anzeigen.

▼ Verarbeitung

▼ listfile

--listfile = `true|false`

Bei `true` wird das Argument *InputFile* des Befehls als Textdatei behandelt, die einen Dateinamen pro Zeile enthält. Der Standardwert ist `false`. (Als Alternative können die Dateien im CLI getrennt durch ein Leerzeichen aufgelistet werden. Beachten Sie allerdings, dass CLIs eine maximale Zeichenanzahl haben). Beachten Sie, dass die Option `--listfile` nur auf Argumente, nicht aber auf Optionen angewendet wird.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

▼ Kataloge und globale Ressourcen

▼ catalog

--catalog = *FILE*

Gibt den absoluten Pfad zu einer Root-Katalog-Datei an, die nicht die installierte Root-Katalog-Datei ist. Der Standardwert ist der absolute Pfad zur installierten Root-Katalog-Datei. (`<installationsordner>\Altova\RaptorXMLServer2025\etc\RootCatalog.xml`). Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge 48.

▼ user-catalog

--user-catalog = *FILE*

Definiert den absoluten Pfad zu einem XML-Katalog, der zusätzlich zum Root-Katalog verwendet werden soll. Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge 48.

▼ enable-globalresources

--enable-globalresources = `true|false`
Aktiviert die globalen Ressourcen[55]. Standardwert ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- globalresourceconfig [gc]

  --gc | --globalresourceconfig = *VALUE*
  Definiert die aktive Konfiguration der globalen Ressource[55] (und aktiviert globale Ressourcen)[55].

- globalresourcefile [gr]

  --gr | --globalresourcefile = *FILE*
  Definiert die globale Ressourcendatei[55] (und aktiviert globale Ressourcen)[55].

Allgemeine Optionen

- error-format

  --error-format = `text|shortxml|longxml`
  Definiert das Format der Fehlerausgabe. Der Standardwert ist `text`. Mit den anderen Optionen werden XML-Formate generiert, wobei mit `longxml` mehr Details generiert werden.

- error-limit

  --error-limit = *N | unlimited*
  Definiert das Fehlerlimit mit einem Wertebereich von `1` bis `9999` oder `unbegrenzt`. Der Standardwert ist `100`. Bei Erreichung des Fehlerlimits wird die Validierung gestoppt. Dient dazu, die Prozessorverwendung während der Validierung/Transformation einzuschränken.

- info-limit

  --info-limit = *N | unlimited*
  Definiert die Grenze für Informationsmeldungen im Bereich von `1-65535` oder als `unlimited` (unbegrenzt). Die Verarbeitung wird auch nach Erreichen des Info-Limits fortgesetzt, doch werden weitere Meldungen nicht mehr ausgegeben. Der Standardwert ist `100`.

- help

  --help
  Zeigt den Hilfetext zum Befehl an. Beispiel: `valany --h`. (Alternativ dazu kann der Befehl `help` zusammen mit einem Argument verwendet werden. Beispiel: `help valany`.)

- listfile

  --listfile = `true|false`
  Bei `true` wird das Argument *InputFile* des Befehls als Textdatei behandelt, die einen Dateinamen pro Zeile enthält. Der Standardwert ist `false`. (Als Alternative können die Dateien im CLI getrennt durch ein Leerzeichen aufgelistet werden. Beachten Sie allerdings, dass CLIs eine maximale Zeichenanzahl haben). Beachten Sie, dass die Option `--listfile` nur auf Argumente, nicht aber auf Optionen angewendet wird.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert

definiert wird.

- log-output

  --log-output = *FILE*
  Schreibt die Meldungsausgabe in die angegebene URL. Stellen Sie sicher, dass das CLI Schreibrechte für den Ausgabepfad hat.

- network-timeout

  --network-timeout = *VALUE*
  Definiert das Timeout für entfernte I/O-Operationen in Millisekunden. Der Standardwert ist: `40000`.

- recurse

  --recurse = `true|false`
  Dient zur Auswahl von Dateien innerhalb von Unterverzeichnissen einschließlich ZIP-Archiven. Bei `true` wählt das Argument *InputFile* des Befehls die angegebene Datei auch in den Unterverzeichnissen aus. Beispiel: `"test.zip|zip\test.xml"` wählt Dateien mit dem Namen `test.xml` auf allen Ordnerebenen des ZIP-Ordners aus. Referenzen auf ZIP-Dateien müssen in Anführungszeichen angegeben werden. Es können die Platzhalter `*` und `?` verwendet werden. Mit `*.xml` werden folglich alle `.xml` Dateien im (ZIP-)Ordner ausgewählt. Der Standardwert der Option ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- verbose

  --verbose = `true|false`
  Mit dem Wert `true` wird die Ausgabe zusätzlicher Informationen bei der Validierung aktiviert. Der Standardwert ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- verbose-output

  --verbose-output = *FILE*
  Schreibt die ausführliche Ausgabe in *FILE*.

- version

  --version
  Zeigt die Version von RaptorXML Server an. Setzen Sie `--version` bei Verwendung mit einem Befehl vor den Befehl.

- warning-limit

  --warning-limit = *N | unlimited*
  Definiert das Warnungslimit im Bereich von `1-65535` oder `unbegrenzt`. Bei Erreichen dieses Limits wird die Verarbeitung fortgesetzt, doch werden keine weiteren Warnungen mehr ausgegeben. Der Standardwert ist `100`.

5.3 XQuery-Befehle

Die XQuery-Befehle sind:

- xquery 98: zum Ausführen von XQuery-Dokumenten, optional mit einem Input-Dokument
- xqueryupdate 106: zum Ausführen eines XQuery Update über ein XQuery-Dokument und optional mit einem zu aktualisierenden Input-Dokument
- valxquery 115: zum Validieren von XQuery-Dokumenten
- valxqueryupdate 122: zum Validieren eines XQuery (Update)-Dokuments

5.3.1 xquery

Der Befehl **xquery** erhält als einziges Argument eine XQuery-Datei und führt diese mit einer optionalen Input-Datei aus, um eine Ausgabedatei zu erzeugen. Die Input-Datei und die Ausgabedatei sind als Optionen definiert.

```
raptorxml xquery [options] XQuery-File
```

- Das Argument ***XQuery-File*** ist der Pfad und Name der auszuführenden XQuery-Datei.
- Sie können XQuery 1.0 oder 3.0 verwenden. Standardmäßig wird XQuery 3.0 verwendet.

Beispiele

Beispiele für den Befehl **xquery**:

- **raptorxml** xquery --output=c:\Output.xml c:\TestQuery.xq
- **raptorxml** xquery --input=c:\Input.xml --output=c:\Output.xml --param=company:"Altova" --p=date:"2006-01-01" c:\TestQuery.xq
- **raptorxml** xquery --input=c:\Input.xml --output=c:\Output.xml --param=source:"doc('c:\test\books.xml')//book "
- **raptorxml** xquery --output=c:\Output.xml --omit-xml-declaration=false --output-encoding=ASCII c:\TestQuery.xq

▼ Groß- und Kleinschreibung und Schrägstriche in der Befehlszeile

RaptorXML (und **RaptorXMLServer** für Administrator-Befehle) *unter Windows*
raptorxml und **raptorxmlserver** für Administrator-Befehle) *unter Windows und Unix (Linux, Mac)*

* Beachten Sie, dass klein geschriebene Befehle (raptorxml und raptorxmlserver) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (RaptorXML) nur unter Windows und Mac ausgeführt werden.
*Verwenden Sie auf Linux und Mac-Systemen Schrägstriche und auf Windows-Systemen umgekehrte Schrägstriche.

▼ Umgekehrte Schrägstriche, Leerzeichen und Sonderzeichen auf Windows-Systemen

Auf Windows-Systemen: Verwenden Sie bei Vorhandensein von Leerzeichen oder Sonerzeichen in Strings (z.B. in Datei- oder Ordnernamen oder Firmen-, Personen- oder Produktnamen) Anführungszeichen: z.B., **"Meine Datei"**. Beachten Sie jedoch, dass ein von einem doppelten Anführungszeichen gefolgter umgekehrter Schrägstrich (z.B: **"C:\Mein Verzeichnis\"**) eventuell nicht

korrekt gelesen wird, da der umgekehrte Schrägstrich auch den Beginn einer Escape-Sequenz markiert und die Escape-Sequenz **\"** für ein doppeltes Anführungszeichen steht. Wenn Sie diese Zeichensequenz mit einem Escape versehen wollen, verwenden Sie einen vorangestellten umgekehrten Schrägstrich, wie den folgenden: **\\"**. Einfacher ausgedrückt: Wenn Sie einen Dateipfad, der Leerzeichen oder einen umgekehrten Schrägstrich am Ende enthält, schreiben müssen, so schreiben Sie diesen folgendermaßen: **"C:\Mein Verzeichnis\\"**.

Optionen

Optionen werden in ihrer kurzen Form (falls verfügbar) und in ihrer langen Form aufgelistet. Für die kurze und die lange Form können ein oder zwei Bindestriche verwendet werden. Eine Option kann, muss aber keinen Wert erhalten. Eine Option, die einen Wert erhält, wird folgendermaßen geschrieben: **--option=wert**. Werte können außer in zwei Fällen ohne Anführungszeichen definiert werden: (i) wenn der Wertestring Leerzeichen enthält oder (ii) wenn in der Beschreibung der Option explizit erwähnt ist, dass Anführungszeichen zwingend erforderlich sind. Wenn eine Option einen Booleschen Wert erhält und kein Wert definiert ist, so ist der Standardwert der Option `TRUE`. Mit Hilfe der Option **--h, --help** können Sie Informationen über den Befehl anzeigen.

- XQuery-Verarbeitung
  - indent-characters

    --indent-characters = *VALUE*
    Definiert den Zeichenstring, der als Einrückung verwendet werden soll.

  - input

    --input = *FILE*
    Die URL der zu transformierenden XML-Datei.

  - omit-xml-declaration

    --omit-xml-declaration = `true|false`
    Serialisierungsoption, mit der angegeben wird, ob die XML-Deklaration in der Ausgabe enthalten sein soll oder nicht. Wenn der Wert `true` ist, enthält das Ausgabedokument keine XML-Deklaration. Wenn der Wert `false` ist, wird eine XML-Deklaration inkludiert. Der Standardwert ist `false`.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

  - output, xsltoutput

    output = *FILE*, **xsltoutput =** *FILE*
    Die URL der primären Ausgabedatei. So ist z.B. im Fall der Ausgabe mehrerer HTML-Dateien die primäre Ausgabedatei der Pfad der Eintrittspunkt-HTML-Datei. Zusätzliche Ausgabedateien wie z.B. generierte Bilddateien werden als `xslt-additional-output-files` angegeben. Wenn keine `--output` oder `--xsltoutput` Option definiert ist, wird die Ausgabe in die Standardausgabe geschrieben.

  - output-encoding

    --output-encoding = *VALUE*
    Der Wert des Kodierungsattributs im Ausgabedokument. Gültige Werte sind die Namen im IANA-

Zeichensatz-Register. Der Standardwert ist `UTF-8`.

- output-indent

  `--output-indent =` `true|false`
  Wenn der Wert `true` ist, wird die Ausgabe entsprechend ihrer hierarchischen Struktur eingerückt. Bei `false` gibt es keine hierarchische Einrückung. Der Standardwert ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- output-method

  `--output-method =` `xml|html|xhtml|text`
  Definiert das Ausgabeformat. Der Standardwert ist `xml`.

- param [p]

  `--p | --param =` *`KEY:VALUE`*

  - ***XQuery***

    Definiert den Wert eines externen Parameters. Ein externer Parameter ist im XQuery-Dokument mit der `declare variable` Deklaration gefolgt von einem Variablennamen und anschließend dem Schlüsselwort `external`, gefolgt von einem Semikolon deklariert. Beispiel:

    ```
    declare variable $foo as xs:string external;
    ```

    Aufgrund des Schlüsselworts `external` wird `$foo` zu einem externen Parameter, dessen Wert zur Laufzeit von einer externen Quelle aus übergeben wird. Der externe Parameter erhält mit dem CLI-Befehl einen Wert. Beispiel:

    ```
    --param=foo:'MyName'
    ```

    In der obigen Beschreibungsanweisung ist *`KEY`* der Name des externen Parameters, *`VALUE`* der als XPath-Ausdruck angegebene Wert des externen Parameters. Im CLI verwendete Parameter müssen im XQuery-Dokument deklariert sein. Wenn mehrere externe Parameter als Werte an das CLI übergeben werden, muss jeder eine separate `--param` Option erhalten. Wenn der XPath-Ausdruck Leerzeichen enthält, muss er in doppelte Anführungszeichen gesetzt werden.

  - ***XSLT***

    Definiert einen globalen Stylesheet-Parameter. *`KEY`* ist der Parametername, *`VALUE`* der als XPath-Ausdruck angegebene Parameterwert. Im CLI verwendete Parameter müssen im Stylesheet deklariert sein. Wenn mehrere Parameter verwendet werden, muss vor jedem Parameter die `--param` Option verwendet werden. Wenn der XPath-Ausdruck Leerzeichen enthält - ob im XPath-Ausdruck selbst oder in einem String-Literal im Ausdruck - muss er in doppelte Anführungszeichen gesetzt werden. Beispiel:

    ```
    raptorxml xslt --input=c:\Test.xml --output=c:\Output.xml --
    param=date://node[1]/@att1 --p=title:'stringwithoutspace' --
    param=title:"'string with spaces'" --p=amount:456 c:\Test.xslt
    ```

- xpath-static-type-errors-as-warnings

  `--xpath-static-type-errors-as-warnings =` `true|false`
  Bei `true` werden alle im statischen XPath-Kontext gefundenen Typ-Fehler auf eine Warnung

herabgestuft. Während die Ausführung bei einem Fehler fehlschlagen würde, könnte sie bei einer Warnung fortgesetzt werden. Die Standardeinstellung ist `false`.

- xquery-version

  `--xquery-version =` `1|1.0|3|3.0|3.1`
  Gibt an, ob der XQuery-Prozessor XQuery 1.0 oder XQuery 3.0 verwenden soll. Der Standardwert ist 3.1.

- XML-Schema und XML-Instanz
  - load-xml-with-psvi

    `--load-xml-with-psvi =` `true|false`
    Ermöglicht die Validierung von XML-Input-Dateien und die Generierung von Informationen für diese Dateien nach Validierung des Schemas. Der Standardwert ist: `true`.

  - schema-imports

    `--schema-imports =` `load-by-schemalocation | load-preferring-schemalocation | load-by-namespace | load-combining-both | license-namespace-only`
    Definiert das Verhalten von `xs:import` Elementen, von denen jedes ein optionales `namespace` Attribut und ein optionales `schemaLocation` Attribut hat: `<import namespace="someNS" schemaLocation="someURL">`. Mit der Option wird definiert, ob ein Schema-Dokument geladen oder nur ein Namespace lizenziert werden soll und, wenn ein Schema-Dokument geladen werden soll, anhand welcher Informationen es gesucht werden soll. Standardeinstellung: `load-preferring-schemalocation`. Das Verhalten ist das folgende:

    - `load-by-schemalocation`: Anhand des Werts des `schemaLocation` Attributs wird der Schemapfad ermittelt, wobei Katalog-Mappings[48] berücksichtigt werden. Wenn das Namespace-Attribut vorhanden ist, wird der Namespace importiert (lizenziert).
    - `load-preferring-schemalocation`: Wenn das `schemaLocation` Attribut vorhanden ist, wird es verwendet, wobei Katalog-Mappings[48] berücksichtigt werden. Falls kein `schemaLocation` Attribut vorhanden ist, wird der Wert des `namespace` Attributs über ein Katalog-Mapping[48] verwendet. Dies ist der **Standardwert**.
    - `load-by-namespace`: Anhand des Werts des `namespace` Attributs wird der Schemapfad über ein Katalog-Mapping[48] ermittelt.
    - `load-combining-both`: Wenn entweder das Attribut `namespace` oder das Attribut `schemaLocation` ein Katalog-Mapping[48] hat, so wird das Mapping verwendet. Wenn beide Attribute Katalog-Mappings[48] haben, ist es vom Wert der Option `--schema-mapping` (XML/XSD-Option[255]) abhängig, welches Mapping verwendet wird. Falls kein Katalog-Mapping[48] vorhanden ist, wird das `schemaLocation` Attribut verwendet.
    - `license-namespace-only`: Der Namespace wird importiert. Kein Schema-Dokument wird importiert.

  - schema-location-hints

    `--schema-location-hints =` `load-by-schemalocation | load-by-namespace | load-combining-both | ignore`
    Definiert das Verhalten der Attribute `xsi:schemaLocation` und `xsi:noNamespaceSchemaLocation`: Legt fest, ob ein Schema-Dokument geladen werden soll und falls ja, anhand welcher Informationen es gesucht werden soll. Standardeinstellung: `load-by-schemalocation`.

- Der Wert `load-by-schemalocation` verwendet die URL des Schemapfads[422] in den Attributen `xsi:schemaLocation` und `xsi:noNamespaceSchemaLocation` in XML-Instanzdokumenten. Dies ist der **Standardwert**.
- Der Wert `load-by-namespace` verwendet den Namespace-Teil[422] von `xsi:schemaLocation` und - im Fall von `xsi:noNamespaceSchemaLocation` - einen leeren String und ermittelt das Schema über ein Katalog-Mapping[48].
- Bei Verwendung von `load-combining-both` und wenn entweder der Namespace-Teil oder der URL-Teil ein Katalog-Mapping[48] hat, so wird das Katalog-Mapping[48] verwendet. Wenn beide Katalog-Mappings[48] haben, ist es vom Wert der `--schema-mapping` Option (XML/XSD-Option[255]) abhängig, welches Mapping verwendet wird. Wenn weder der Namespace noch die URL ein Katalog-Mapping hat, wird die URL verwendet.
- Wenn der Wert der Option `ignore` ist, werden die beiden Attribute `xsi:schemaLocation` und `xsi:noNamespaceSchemaLocation` ignoriert.

▼ schema-mapping

`--schema-mapping =` `prefer-schemalocation | prefer-namespace`
Wenn ein Schema-Dokument sowohl anhand des Schema-Pfads als auch des Namespace gesucht werden soll, wird damit festgelegt, welche der beiden Optionen beim Katalog-Lookup Vorrang erhält. (Wenn eine der Optionen `--schemalocation-hints` oder `--schema-imports` einen Wert `load-combining-both` hat und wenn die betroffenen Namespace- und URL-Teile beide Katalog-Mappings[48] haben, gibt der Wert dieser Option an, welches der beiden Mappings verwendet werden soll (das Namespace Mapping oder das URL-Mapping; der Wert `prefer-schemalocation` bezieht sich auf das URL-Mapping)). Der Standardwert ist `prefer-schemalocation`.

▼ xinclude

`--xinclude =` `true|false`
Aktiviert die Unterstützung für XML-Inkludierungen (XInclude). Der Standardwert ist `false`. Bei `false` werden die `include`-Elemente von XInclude ignoriert.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

▼ xml-mode

`--xml-mode =` `wf|id|valid`
Definiert, welcher XML-Verarbeitungsmodus für das XML-Instanzdokument verwendet werden soll: `wf`=Wohlgeformtheitsprüfung; `id`=Wohlgeformtheitsprüfung mit ID/IDREF-Prüfung; `valid`=Validierung. Der Standardwert ist `wf`. Beachten Sie, dass jedes bei der Verarbeitung geladene Instanzdokument eine DTD referenzieren muss, damit der Wert `valid` ausgegeben werden kann. Falls keine DTD vorhanden ist, wird ein Fehler ausgegeben.

▼ xml-mode-for-schemas

`--xml-mode-for-schemas =` `wf|id|valid`
Definiert, welcher XML-Verarbeitungsmodus für XML-Schema-Dokumente verwendet werden soll: `wf`=Wohlgeformtheitsprüfung; `id`=Wohlgeformtheitsprüfung mit ID/IDREF-Prüfung; `valid`=Validierung. Der Standardwert ist `wf`. Beachten Sie, dass jedes bei der Verarbeitung geladene Schema-Dokument eine DTD referenzieren muss, damit der Wert `valid` ausgegeben werden kann. Falls keine DTD vorhanden ist, wird ein Fehler ausgegeben.

▼ xml-validation-error-as-warning

`--xml-validation-error-as-warning =` `true|false`
Bei `true` werden Validierungsfehler als Warnungen behandelt. Wenn Fehler als Warnungen

behandelt werden, wird die weitere Verarbeitung, z.B. eine XSLT-Transformation ungeachtet der Fehler fortgesetzt. Die Standardeinstellung ist `false`.

- xpath-static-type-errors-as-warnings

  `--xpath-static-type-errors-as-warnings =` `true|false`
  Bei `true` werden alle im statischen XPath-Kontext gefundenen Typ-Fehler auf eine Warnung herabgestuft. Während die Ausführung bei einem Fehler fehlschlagen würde, könnte sie bei einer Warnung fortgesetzt werden. Die Standardeinstellung ist `false`.

- xsd

  `--xsd =` *FILE*
  Definiert ein oder mehrere XML-Schema-Dokumente, die für die Validierung von XML-Instanzdokumenten verwendet werden sollen. Um mehr als ein Schema-Dokument zu definieren, fügen Sie die Option mehrmals hinzu.

- xsd-version

  `--xsd-version =` `1.0|1.1|detect`
  Definiert die zu verwendende W3C Schema Definition Language (XSD) Version. Der Standardwert ist `1.0`. Diese Option eignet sich auch, um herauszufinden, inwiefern ein 1.0-kompatibles Schema nicht mit Schemaversion 1.1 kompatibel ist. Die Option `detect` ist eine Altova-spezifische Funktionalität. Mit dieser Option kann die Version des XML-Schema-Dokuments (`1.0` oder `1.1`) durch Lesen des Werts des `vc:minVersion` Attributs des `<xs:schema>` Elements des Dokuments ermittelt werden. Wenn der Wert des `@vc:minVersion` Attributs `1.1` ist, wird das Schema als Version `1.1` erkannt. Bei jedem anderen Wert wird das Schema als `1.0` erkannt. Bei jedem anderen Wert oder bei Fehlen des `@vc:minVersion` Attributs wird das Schema als Version `1.0` gelesen.

- Kataloge und globale Ressourcen
  - catalog

    `--catalog =` *FILE*
    Gibt den absoluten Pfad zu einer Root-Katalog-Datei an, die nicht die installierte Root-Katalog-Datei ist. Der Standardwert ist der absolute Pfad zur installierten Root-Katalog-Datei. (`<installationsordner>\Altova\RaptorXMLServer2025\etc\RootCatalog.xml`). Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge 48.

  - user-catalog

    `--user-catalog =` *FILE*
    Definiert den absoluten Pfad zu einem XML-Katalog, der zusätzlich zum Root-Katalog verwendet werden soll. Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge 48.

  - enable-globalresources

    `--enable-globalresources =` `true|false`
    Aktiviert die globalen Ressourcen 55. Standardwert ist `false`.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

  - globalresourceconfig [gc]

--gc | --globalresourceconfig = *VALUE*
Definiert die aktive Konfiguration der globalen Ressource[55] (und aktiviert globale Ressourcen)[55].

- globalresourcefile [gr]

  --gr | --globalresourcefile = *FILE*
  Definiert die globale Ressourcendatei[55] (und aktiviert globale Ressourcen)[55].

- Erweiterungen

  Diese Optionen definieren die Behandlung von speziellen Erweiterungsfunktionen, die in einer Reihe von Enterprise Versionen von Altova-Produkten (wie z.B. in XMLSpy Enterprise Edition) verfügbar sind. Die Verwendung dieser Funktionen ist im Benutzerhandbuch des jeweiligen Produkts beschrieben.

  - chartext-disable

    --chartext-disable = `true|false`
    Deaktiviert Diagrammerweiterungen. Der Standardwert ist `false`.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

  - dotnetext-disable

    --dotnetext-disable = `true|false`
    Deaktiviert .NET-Erweiterungen. Der Standardwert ist `false`.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

  - jvm-location

    --jvm-location = *FILE*
    FILE definiert den Pfad zur Java Virtual Machine (DLL unter Windows, freigegebenes Objekt unter Linux). Sie benötigen JVM, wenn Sie Java-Erweiterungsfunktionen[530] in Ihrem XSLT/XQuery-Code verwenden. Die Standardeinstellung ist `false`.

  - javaext-barcode-location

    --javaext-barcode-location = *FILE*
    Definiert den Pfad zum Ordner, der die Barcode-Erweiterungsdatei `AltovaBarcodeExtension.jar` enthält. Der Pfad muss in einer der folgenden Formen angegeben werden:

    - als Datei-URI, z.B.: `--javaext-barcode-location="file:///C:/Program Files/Altova/RaptorXMLServer2025/etc/jar/"`
    - als Windows-Pfad mit maskierten umgekehrten Schrägstrichen, z.B.: `--javaext-barcode-location="C:\\Program Files\\Altova\\RaptorXMLServer2025\\etc\\jar\\"`

  - javaext-disable

    --javaext-disable = `true|false`
    Deaktiviert Java-Erweiterungen. Der Standardwert ist `false`.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- Allgemeine Optionen
  - error-format

    `--error-format =` `text|shortxml|longxml`
    Definiert das Format der Fehlerausgabe. Der Standardwert ist `text`. Mit den anderen Optionen werden XML-Formate generiert, wobei mit `longxml` mehr Details generiert werden.

  - error-limit

    `--error-limit =` *`N | unlimited`*
    Definiert das Fehlerlimit mit einem Wertebereich von `1` bis `9999` oder `unbegrenzt`. Der Standardwert ist `100`. Bei Erreichung des Fehlerlimits wird die Validierung gestoppt. Dient dazu, die Prozessorverwendung während der Validierung/Transformation einzuschränken.

  - info-limit

    `--info-limit =` *`N | unlimited`*
    Definiert die Grenze für Informationsmeldungen im Bereich von `1-65535` oder als `unlimited` (unbegrenzt). Die Verarbeitung wird auch nach Erreichen des Info-Limits fortgesetzt, doch werden weitere Meldungen nicht mehr ausgegeben. Der Standardwert ist `100`.

  - help

    `--help`
    Zeigt den Hilfetext zum Befehl an. Beispiel: `valany --h`. (Alternativ dazu kann der Befehl `help` zusammen mit einem Argument verwendet werden. Beispiel: `help valany`.)

  - listfile

    `--listfile =` `true|false`
    Bei `true` wird das Argument *`InputFile`* des Befehls als Textdatei behandelt, die einen Dateinamen pro Zeile enthält. Der Standardwert ist `false`. (Als Alternative können die Dateien im CLI getrennt durch ein Leerzeichen aufgelistet werden. Beachten Sie allerdings, dass CLIs eine maximale Zeichenanzahl haben). Beachten Sie, dass die Option `--listfile` nur auf Argumente, nicht aber auf Optionen angewendet wird.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

  - log-output

    `--log-output =` *`FILE`*
    Schreibt die Meldungsausgabe in die angegebene URL. Stellen Sie sicher, dass das CLI Schreibrechte für den Ausgabepfad hat.

  - network-timeout

    `--network-timeout =` *`VALUE`*
    Definiert das Timeout für entfernte I/O-Operationen in Millisekunden. Der Standardwert ist: `40000`.

  - recurse

    `--recurse =` `true|false`
    Dient zur Auswahl von Dateien innerhalb von Unterverzeichnissen einschließlich ZIP-Archiven. Bei

`true` wählt das Argument *`InputFile`* des Befehls die angegebene Datei auch in den Unterverzeichnissen aus. Beispiel: `"test.zip|zip\test.xml"` wählt Dateien mit dem Namen `test.xml` auf allen Ordnerebenen des ZIP-Ordners aus. Referenzen auf ZIP-Dateien müssen in Anführungszeichen angegeben werden. Es können die Platzhalter `*` und `?` verwendet werden. Mit `*.xml` werden folglich alle `.xml` Dateien im (ZIP-)Ordner ausgewählt. Der Standardwert der Option ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- verbose

  `--verbose =` `true|false`
  Mit dem Wert `true` wird die Ausgabe zusätzlicher Informationen bei der Validierung aktiviert. Der Standardwert ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- verbose-output

  `--verbose-output =` *`FILE`*
  Schreibt die ausführliche Ausgabe in *`FILE`*.

- version

  `--version`
  Zeigt die Version von RaptorXML Server an. Setzen Sie `--version` bei Verwendung mit einem Befehl vor den Befehl.

- warning-limit

  `--warning-limit =` *`N | unlimited`*
  Definiert das Warnungslimit im Bereich von `1-65535` oder `unbegrenzt`. Bei Erreichen dieses Limits wird die Verarbeitung fortgesetzt, doch werden keine weiteren Warnungen mehr ausgegeben. Der Standardwert ist `100`.

5.3.2 xqueryupdate

Der Befehl **`valxqueryupdate`** erhält als einziges Argument eine XQuery- oder XQuery-Update-Datei und führt diese aus. Wenn eine optionale XML-Input-Datei definiert wird, so wird diese XML-Datei mit den in der ***`XQuery(Update)-Datei`*** bereitgestellten XQuery Update-Befehlen verarbeitet. In diesem Fall können die Aktualisierungen direkt auf die Input-Datei angewendet werden oder die aktualisierten XML-Daten können in eine XML-Ausgabedatei geschrieben werden. Die Input-Datei und die Ausgabedatei sind als Optionen definiert. Wenn die ***`XQuery(Update)-Datei`*** nur XQuery-Anweisungen und keine XQuery Update-Anweisungen enthält, führt der Befehl eine einfache XQuery-Ausführung durch.

`raptorxml xqueryupdate [options]` ***`XQuery(Update)-File`***

- Das Argument ***XQuery(Update)-File*** ist der Pfad und Name der auszuführenden XQuery- (**.xq**) oder XQuery Update- (**.xqu**) Datei. Wenn die Datei XQuery Update-Anweisungen enthält, werden diese an der XML-Input-Datei ausgeführt. Andernfalls fungiert der Befehl als XQuery-Ausführungsbefehl.
- Sie können angeben, ob XQuery Update 1.0 oder 3.0 verwendet werden soll. Standardmäßig wird XQuery Update 3.0 verwendet.

Beispiele

Beispiele für den Befehl **xqueryupdate**:

- **raptorxml** `xqueryupdate --output=c:\Output.xml c:\TestQuery.xq` (Schreibt das Resultat der XQuery-Datei in die Ausgabedatei.)
- **raptorxml** `xqueryupdate --input=c:\Input.xml --output=c:\Output.xml --updated-xml=asmainresult c:\UpdateFile.xqu` (Aktualisiert `Input.xml` anhand der Aktualisierungsanweisungen in `UpdateFile.xqu` und schreibt die Aktualisierung in `Output.xml`.)
- **raptorxml** `xqueryupdate --input=c:\Input.xml --output=c:\Output.xml --updated-xml=writeback c:\UpdateFile.xq` (Aktualisiert `Input.xml` anhand der Aktualisierungsanweisungen in `UpdateFile.xq`. Die Datei `Output.xml` wird nicht erstellt.)
- **raptorxml** `xqueryupdate --input=c:\Input.xml --output=c:\Output.xml --updated-xml=discard c:\TestQuery.xqu` (Die Aktualisierungen werden verworfen. Die Input-Datei wird nicht geändert. Die Datei `Output.xml` wird erstellt, aber enthält keine aktualisierten XML-Daten.)
- **raptorxml** `xqueryupdate --input=c:\Input.xml --output=c:\Output.xml c:\TestQuery.xqu` (Die Aktualisierungen werden wie im vorhergehenden Beispiel verworfen. Der Grund dafür ist, dass der Standardwert der Option `--updated-xml discard` ist.)

▼ Groß- und Kleinschreibung und Schrägstriche in der Befehlszeile

RaptorXML (und **RaptorXMLServer** für Administrator-Befehle) *unter Windows*
raptorxml und **raptorxmlserver** für Administrator-Befehle) *unter Windows und Unix (Linux, Mac)*

* Beachten Sie, dass klein geschriebene Befehle (`raptorxml` und `raptorxmlserver`) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (`RaptorXML`) nur unter Windows und Mac ausgeführt werden.
*Verwenden Sie auf Linux und Mac-Systemen Schrägstriche und auf Windows-Systemen umgekehrte Schrägstriche.

▼ Umgekehrte Schrägstriche, Leerzeichen und Sonderzeichen auf Windows-Systemen

Auf Windows-Systemen: Verwenden Sie bei Vorhandensein von Leerzeichen oder Sonerzeichen in Strings (z.B. in Datei- oder Ordnernamen oder Firmen-, Personen- oder Produktnamen) Anführungszeichen: z.B., **"Meine Datei"**. Beachten Sie jedoch, dass ein von einem doppelten Anführungszeichen gefolgter umgekehrter Schrägstrich (z.B: **"C:\Mein Verzeichnis\"**) eventuell nicht korrekt gelesen wird, da der umgekehrte Schrägstrich auch den Beginn einer Escape-Sequenz markiert und die Escape-Sequenz **\"** für ein doppeltes Anführungszeichen steht. Wenn Sie diese Zeichensequenz mit einem Escape versehen wollen, verwenden Sie einen vorangestellten umgekehrten Schrägstrich, wie den folgenden: **\\"**. Einfacher ausgedrückt: Wenn Sie einen Dateipfad, der Leerzeichen oder einen umgekehrten Schrägstrich am Ende enthält, schreiben müssen, so schreiben Sie diesen folgendermaßen: **"C:\Mein Verzeichnis\\"**.

Optionen

Optionen werden in ihrer kurzen Form (falls verfügbar) und in ihrer langen Form aufgelistet. Für die kurze und

die lange Form können ein oder zwei Bindestriche verwendet werden. Eine Option kann, muss aber keinen Wert erhalten. Eine Option, die einen Wert erhält, wird folgendermaßen geschrieben: **`--option=wert`**. Werte können außer in zwei Fällen ohne Anführungszeichen definiert werden: (i) wenn der Wertestring Leerzeichen enthält oder (ii) wenn in der Beschreibung der Option explizit erwähnt ist, dass Anführungszeichen zwingend erforderlich sind. Wenn eine Option einen Booleschen Wert erhält und kein Wert definiert ist, so ist der Standardwert der Option `TRUE`. Mit Hilfe der Option **`--h, --help`** können Sie Informationen über den Befehl anzeigen.

- XQuery Update-Verarbeitung
  - indent-characters

    `--indent-characters =` *VALUE*
    Definiert den Zeichenstring, der als Einrückung verwendet werden soll.

  - input

    `--input =` *FILE*
    Die URL der zu transformierenden XML-Datei.

  - omit-xml-declaration

    `--omit-xml-declaration =` `true|false`
    Serialisierungsoption, mit der angegeben wird, ob die XML-Deklaration in der Ausgabe enthalten sein soll oder nicht. Wenn der Wert `true` ist, enthält das Ausgabedokument keine XML-Deklaration. Wenn der Wert `false` ist, wird eine XML-Deklaration inkludiert. Der Standardwert ist `false`.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

  - output, xsltoutput

    `output =` *FILE*, **`xsltoutput =`** *FILE*
    Die URL der primären Ausgabedatei. So ist z.B. im Fall der Ausgabe mehrerer HTML-Dateien die primäre Ausgabedatei der Pfad der Eintrittspunkt-HTML-Datei. Zusätzliche Ausgabedateien wie z.B. generierte Bilddateien werden als `xslt-additional-output-files` angegeben. Wenn keine `--output` oder `--xsltoutput` Option definiert ist, wird die Ausgabe in die Standardausgabe geschrieben.

  - output-encoding

    `--output-encoding =` *VALUE*
    Der Wert des Kodierungsattributs im Ausgabedokument. Gültige Werte sind die Namen im IANA-Zeichensatz-Register. Der Standardwert ist `UTF-8`.

  - output-indent

    `--output-indent =` `true|false`
    Wenn der Wert `true` ist, wird die Ausgabe entsprechend ihrer hierarchischen Struktur eingerückt. Bei `false` gibt es keine hierarchische Einrückung. Der Standardwert ist `false`.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

  - output-method

--output-method = xml|html|xhtml|text
Definiert das Ausgabeformat. Der Standardwert ist xml.

- param [p]

--p | --param = *KEY:VALUE*

XQuery

Definiert den Wert eines externen Parameters. Ein externer Parameter ist im XQuery-Dokument mit der declare variable Deklaration gefolgt von einem Variablennamen und anschließend dem Schlüsselwort external, gefolgt von einem Semikolon deklariert. Beispiel:

```
declare variable $foo as xs:string external;
```

Aufgrund des Schlüsselworts external wird $foo zu einem externen Parameter, dessen Wert zur Laufzeit von einer externen Quelle aus übergeben wird. Der externe Parameter erhält mit dem CLI-Befehl einen Wert. Beispiel:

```
--param=foo:'MyName'
```

In der obigen Beschreibungsanweisung ist *KEY* der Name des externen Parameters, *VALUE* der als XPath-Ausdruck angegebene Wert des externen Parameters. Im CLI verwendete Parameter müssen im XQuery-Dokument deklariert sein. Wenn mehrere externe Parameter als Werte an das CLI übergeben werden, muss jeder eine separate --param Option erhalten. Wenn der XPath-Ausdruck Leerzeichen enthält, muss er in doppelte Anführungszeichen gesetzt werden.

XSLT

Definiert einen globalen Stylesheet-Parameter. *KEY* ist der Parametername, *VALUE* der als XPath-Ausdruck angegebene Parameterwert. Im CLI verwendete Parameter müssen im Stylesheet deklariert sein. Wenn mehrere Parameter verwendet werden, muss vor jedem Parameter die --param Option verwendet werden. Wenn der XPath-Ausdruck Leerzeichen enthält - ob im XPath-Ausdruck selbst oder in einem String-Literal im Ausdruck - muss er in doppelte Anführungszeichen gesetzt werden. Beispiel:

```
raptorxml xslt --input=c:\Test.xml --output=c:\Output.xml --
param=date://node[1]/@att1 --p=title:'stringwithoutspace' --
param=title:"'string with spaces'" --p=amount:456 c:\Test.xslt
```

- xpath-static-type-errors-as-warnings

--xpath-static-type-errors-as-warnings = true|false
Bei true werden alle im statischen XPath-Kontext gefundenen Typ-Fehler auf eine Warnung herabgestuft. Während die Ausführung bei einem Fehler fehlschlagen würde, könnte sie bei einer Warnung fortgesetzt werden. Die Standardeinstellung ist false.

- xquery-update-version

--xquery-update-version = 1|1.0|3|3.0|
Definiert, ob der XQuery-Prozessor XQuery Update Facility 1.0 oder XQuery Update Facility 3.0 verwenden soll. Der Standardwert ist 3.

- keep-formatting

--keep-formatting = true|false

Behält die Formatierung des Zieldokuments so gut wie möglich bei. Der Standardwert ist: `true`.

- updated-xml

  `--updated-xml =` `discard|writeback|asmainresult`
  Definiert, wie die aktualisierte XML-Datei behandelt werden soll.

  - `discard`: Die Aktualisierung wird verworfen und nicht in eine Datei geschrieben. Weder die Input-Datei noch die Output-Datei wird aktualisiert. Beachten Sie, dass dies die Standardeinstellung ist.
  - `writeback`: Schreibt die Aktualisierung zurück in die mit der Option `--input` definierte XML-Input-Datei.
  - `asmainresult`: Schreibt die Aktualisierung in die mit der Option `--output` definierte XML-Output-Datei. Wenn die **`--output`** Option nicht definiert wurde, wird die Aktualisierung in die Standardausgabe geschrieben. In beiden Fällen wird die XML-Input-Datei nicht geändert.

  Der Standardwert ist `discard`.

- XML-Schema und XML-Instanz

  - load-xml-with-psvi

    `--load-xml-with-psvi =` `true|false`
    Ermöglicht die Validierung von XML-Input-Dateien und die Generierung von Informationen für diese Dateien nach Validierung des Schemas. Der Standardwert ist: `true`.

  - schema-imports

    `--schema-imports =` `load-by-schemalocation | load-preferring-schemalocation | load-by-namespace | load-combining-both | license-namespace-only`
    Definiert das Verhalten von `xs:import` Elementen, von denen jedes ein optionales `namespace` Attribut und ein optionales `schemaLocation` Attribut hat: `<import namespace="someNS" schemaLocation="someURL">`. Mit der Option wird definiert, ob ein Schema-Dokument geladen oder nur ein Namespace lizenziert werden soll und, wenn ein Schema-Dokument geladen werden soll, anhand welcher Informationen es gesucht werden soll. Standardeinstellung: `load-preferring-schemalocation`. Das Verhalten ist das folgende:

    - `load-by-schemalocation`: Anhand des Werts des `schemaLocation` Attributs wird der Schemapfad ermittelt, wobei Katalog-Mappings[48] berücksichtigt werden. Wenn das Namespace-Attribut vorhanden ist, wird der Namespace importiert (lizenziert).
    - `load-preferring-schemalocation`: Wenn das `schemaLocation` Attribut vorhanden ist, wird es verwendet, wobei Katalog-Mappings[48] berücksichtigt werden. Falls kein `schemaLocation` Attribut vorhanden ist, wird der Wert des `namespace` Attributs über ein Katalog-Mapping[48] verwendet. Dies ist der **Standardwert**.
    - `load-by-namespace`: Anhand des Werts des `namespace` Attributs wird der Schemapfad über ein Katalog-Mapping[48] ermittelt.
    - `load-combining-both`: Wenn entweder das Attribut `namespace` oder das Attribut `schemaLocation` ein Katalog-Mapping[48] hat, so wird das Mapping verwendet. Wenn beide Attribute Katalog-Mappings[48] haben, ist es vom Wert der Option `--schema-mapping` (XML/XSD-Option[255]) abhängig, welches Mapping verwendet wird. Falls kein Katalog-Mapping[48] vorhanden ist, wird das `schemaLocation` Attribut verwendet.
    - `license-namespace-only`: Der Namespace wird importiert. Kein Schema-Dokument wird

importiert.

- schema-location-hints

  --schema-location-hints = `load-by-schemalocation | load-by-namespace | load-combining-both | ignore`
  Definiert das Verhalten der Attribute `xsi:schemaLocation` und `xsi:noNamespaceSchemaLocation`: Legt fest, ob ein Schema-Dokument geladen werden soll und falls ja, anhand welcher Informationen es gesucht werden soll. Standardeinstellung: `load-by-schemalocation`.

  - Der Wert `load-by-schemalocation` verwendet die URL des Schemapfads[422] in den Attributen `xsi:schemaLocation` und `xsi:noNamespaceSchemaLocation` in XML-Instanzdokumenten. Dies ist der **Standardwert**.
  - Der Wert `load-by-namespace` verwendet den Namespace-Teil[422] von `xsi:schemaLocation` und - im Fall von `xsi:noNamespaceSchemaLocation` - einen leeren String und ermittelt das Schema über ein Katalog-Mapping[48].
  - Bei Verwendung von `load-combining-both` und wenn entweder der Namespace-Teil oder der URL-Teil ein Katalog-Mapping[48] hat, so wird das Katalog-Mapping[48] verwendet. Wenn beide Katalog-Mappings[48] haben, ist es vom Wert der `--schema-mapping` Option (XML/XSD-Option[255]) abhängig, welches Mapping verwendet wird. Wenn weder der Namespace noch die URL ein Katalog-Mapping hat, wird die URL verwendet.
  - Wenn der Wert der Option `ignore` ist, werden die beiden Attribute `xsi:schemaLocation` und `xsi:noNamespaceSchemaLocation` ignoriert.

- schema-mapping

  --schema-mapping = `prefer-schemalocation | prefer-namespace`
  Wenn ein Schema-Dokument sowohl anhand des Schema-Pfads als auch des Namespace gesucht werden soll, wird damit festgelegt, welche der beiden Optionen beim Katalog-Lookup Vorrang erhält. (Wenn eine der Optionen `--schemalocation-hints` oder `--schema-imports` einen Wert `load-combining-both` hat und wenn die betroffenen Namespace- und URL-Teile beide Katalog-Mappings[48] haben, gibt der Wert dieser Option an, welches der beiden Mappings verwendet werden soll (das Namespace Mapping oder das URL-Mapping; der Wert `prefer-schemalocation` bezieht sich auf das URL-Mapping)). Der Standardwert ist `prefer-schemalocation`.

- xinclude

  --xinclude = `true|false`
  Aktiviert die Unterstützung für XML-Inkludierungen (XInclude). Der Standardwert ist `false`. Bei `false` werden die `include`-Elemente von XInclude ignoriert.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- xml-mode

  --xml-mode = `wf|id|valid`
  Definiert, welcher XML-Verarbeitungsmodus für das XML-Instanzdokument verwendet werden soll: `wf`=Wohlgeformtheitsprüfung; `id`=Wohlgeformtheitsprüfung mit ID/IDREF-Prüfung; `valid`=Validierung. Der Standardwert ist `wf`. Beachten Sie, dass jedes bei der Verarbeitung geladene Instanzdokument eine DTD referenzieren muss, damit der Wert `valid` ausgegeben werden kann. Falls keine DTD vorhanden ist, wird ein Fehler ausgegeben.

- xml-mode-for-schemas

--xml-mode-for-schemas = wf|id|valid
Definiert, welcher XML-Verarbeitungsmodus für XML-Schema-Dokumente verwendet werden soll: wf=Wohlgeformtheitsprüfung; id=Wohlgeformtheitsprüfung mit ID/IDREF-Prüfung; valid=Validierung. Der Standardwert ist wf. Beachten Sie, dass jedes bei der Verarbeitung geladene Schema-Dokument eine DTD referenzieren muss, damit der Wert valid ausgegeben werden kann. Falls keine DTD vorhanden ist, wird ein Fehler ausgegeben.

- xml-validation-error-as-warning

--xml-validation-error-as-warning = true|false
Bei true werden Validierungsfehler als Warnungen behandelt. Wenn Fehler als Warnungen behandelt werden, wird die weitere Verarbeitung, z.B. eine XSLT-Transformation ungeachtet der Fehler fortgesetzt. Die Standardeinstellung ist false.

- xsd

--xsd = *FILE*
Definiert ein oder mehrere XML-Schema-Dokumente, die für die Validierung von XML-Instanzdokumenten verwendet werden sollen. Um mehr als ein Schema-Dokument zu definieren, fügen Sie die Option mehrmals hinzu.

- xsd-version

--xsd-version = 1.0|1.1|detect
Definiert die zu verwendende W3C Schema Definition Language (XSD) Version. Der Standardwert ist 1.0. Diese Option eignet sich auch, um herauszufinden, inwiefern ein 1.0-kompatibles Schema nicht mit Schemaversion 1.1 kompatibel ist. Die Option detect ist eine Altova-spezifische Funktionalität. Mit dieser Option kann die Version des XML-Schema-Dokuments (1.0 oder 1.1) durch Lesen des Werts des vc:minVersion Attributs des <xs:schema> Elements des Dokuments ermittelt werden. Wenn der Wert des @vc:minVersion Attributs 1.1 ist, wird das Schema als Version 1.1 erkannt. Bei jedem anderen Wert wird das Schema als 1.0 erkannt. Bei jedem anderen Wert oder bei Fehlen des @vc:minVersion Attributs wird das Schema als Version 1.0 gelesen.

- Kataloge und globale Ressourcen
  - catalog

--catalog = *FILE*
Gibt den absoluten Pfad zu einer Root-Katalog-Datei an, die nicht die installierte Root-Katalog-Datei ist. Der Standardwert ist der absolute Pfad zur installierten Root-Katalog-Datei. (<installationsordner>\Altova\RaptorXMLServer2025\etc\RootCatalog.xml). Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge 48.

  - user-catalog

--user-catalog = *FILE*
Definiert den absoluten Pfad zu einem XML-Katalog, der zusätzlich zum Root-Katalog verwendet werden soll. Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge 48.

  - enable-globalresources

--enable-globalresources = true|false
Aktiviert die globalen Ressourcen 55. Standardwert ist false.
Hinweis: Die Booleschen Optionswerte werden auf true gesetzt, wenn die Option ohne einen Wert

definiert wird.

- globalresourceconfig [gc]

  --gc | --globalresourceconfig = *VALUE*
  Definiert die aktive Konfiguration der globalen Ressource[55] (und aktiviert globale Ressourcen)[55].

- globalresourcefile [gr]

  --gr | --globalresourcefile = *FILE*
  Definiert die globale Ressourcendatei[55] (und aktiviert globale Ressourcen)[55].

- Erweiterungen

  Diese Optionen definieren die Behandlung von speziellen Erweiterungsfunktionen, die in einer Reihe von Enterprise Versionen von Altova-Produkten (wie z.B. in XMLSpy Enterprise Edition) verfügbar sind. Die Verwendung dieser Funktionen ist im Benutzerhandbuch des jeweiligen Produkts beschrieben.

  - chartext-disable

    --chartext-disable = `true|false`
    Deaktiviert Diagrammerweiterungen. Der Standardwert ist `false`.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

  - dotnetext-disable

    --dotnetext-disable = `true|false`
    Deaktiviert .NET-Erweiterungen. Der Standardwert ist `false`.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

  - jvm-location

    --jvm-location = *FILE*
    FILE definiert den Pfad zur Java Virtual Machine (DLL unter Windows, freigegebenes Objekt unter Linux). Sie benötigen JVM, wenn Sie Java-Erweiterungsfunktionen[530] in Ihrem XSLT/XQuery-Code verwenden. Die Standardeinstellung ist `false`.

  - javaext-barcode-location

    --javaext-barcode-location = *FILE*
    Definiert den Pfad zum Ordner, der die Barcode-Erweiterungsdatei `AltovaBarcodeExtension.jar` enthält. Der Pfad muss in einer der folgenden Formen angegeben werden:

    - als Datei-URI, z.B.: `--javaext-barcode-location="file:///C:/Program Files/Altova/RaptorXMLServer2025/etc/jar/"`
    - als Windows-Pfad mit maskierten umgekehrten Schrägstrichen, z.B.: `--javaext-barcode-location="C:\\Program Files\\Altova\\RaptorXMLServer2025\\etc\\jar\\"`

  - javaext-disable

    --javaext-disable = `true|false`
    Deaktiviert Java-Erweiterungen. Der Standardwert ist `false`.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert

definiert wird.

- Allgemeine Optionen
  - error-format

    --error-format = `text|shortxml|longxml`
    Definiert das Format der Fehlerausgabe. Der Standardwert ist `text`. Mit den anderen Optionen werden XML-Formate generiert, wobei mit `longxml` mehr Details generiert werden.

  - error-limit

    --error-limit = *N | unlimited*
    Definiert das Fehlerlimit mit einem Wertebereich von `1` bis `9999` oder `unbegrenzt`. Der Standardwert ist `100`. Bei Erreichung des Fehlerlimits wird die Validierung gestoppt. Dient dazu, die Prozessorverwendung während der Validierung/Transformation einzuschränken.

  - info-limit

    --info-limit = *N | unlimited*
    Definiert die Grenze für Informationsmeldungen im Bereich von `1-65535` oder als `unlimited` (unbegrenzt). Die Verarbeitung wird auch nach Erreichen des Info-Limits fortgesetzt, doch werden weitere Meldungen nicht mehr ausgegeben. Der Standardwert ist `100`.

  - help

    --help
    Zeigt den Hilfetext zum Befehl an. Beispiel: `valany --h`. (Alternativ dazu kann der Befehl `help` zusammen mit einem Argument verwendet werden. Beispiel: `help valany`.)

  - listfile

    --listfile = `true|false`
    Bei `true` wird das Argument *InputFile* des Befehls als Textdatei behandelt, die einen Dateinamen pro Zeile enthält. Der Standardwert ist `false`. (Als Alternative können die Dateien im CLI getrennt durch ein Leerzeichen aufgelistet werden. Beachten Sie allerdings, dass CLIs eine maximale Zeichenanzahl haben). Beachten Sie, dass die Option `--listfile` nur auf Argumente, nicht aber auf Optionen angewendet wird.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

  - log-output

    --log-output = *FILE*
    Schreibt die Meldungsausgabe in die angegebene URL. Stellen Sie sicher, dass das CLI Schreibrechte für den Ausgabepfad hat.

  - network-timeout

    --network-timeout = *VALUE*
    Definiert das Timeout für entfernte I/O-Operationen in Millisekunden. Der Standardwert ist: `40000`.

- recurse

  --recurse = `true|false`
  Dient zur Auswahl von Dateien innerhalb von Unterverzeichnissen einschließlich ZIP-Archiven. Bei `true` wählt das Argument *`InputFile`* des Befehls die angegebene Datei auch in den Unterverzeichnissen aus. Beispiel: `"test.zip|zip\test.xml"` wählt Dateien mit dem Namen `test.xml` auf allen Ordnerebenen des ZIP-Ordners aus. Referenzen auf ZIP-Dateien müssen in Anführungszeichen angegeben werden. Es können die Platzhalter `*` und `?` verwendet werden. Mit `*.xml` werden folglich alle `.xml` Dateien im (ZIP-)Ordner ausgewählt. Der Standardwert der Option ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- verbose

  --verbose = `true|false`
  Mit dem Wert `true` wird die Ausgabe zusätzlicher Informationen bei der Validierung aktiviert. Der Standardwert ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- verbose-output

  --verbose-output = *`FILE`*
  Schreibt die ausführliche Ausgabe in *`FILE`*.

- version

  --version
  Zeigt die Version von RaptorXML Server an. Setzen Sie `--version` bei Verwendung mit einem Befehl vor den Befehl.

- warning-limit

  --warning-limit = *`N | unlimited`*
  Definiert das Warnungslimit im Bereich von `1-65535` oder `unbegrenzt`. Bei Erreichen dieses Limits wird die Verarbeitung fortgesetzt, doch werden keine weiteren Warnungen mehr ausgegeben. Der Standardwert ist `100`.

5.3.3 valxquery

Der Befehl **`valxquery`** erhält als einziges Argument eine XQuery-Datei und validiert diese.

```
raptorxml valxquery [options] XQuery-File
```

- Das Argument ***`XQuery-File`*** ist der Pfad und Name der zu validierenden XQuery-Datei.

Beispiele

Beispiele für den Befehl **valxquery**:

- **raptorxml** valxquery c:\Test.xquery
- **raptorxml** valxquery --xquery-version=1 c:\Test.xquery

▼ Groß- und Kleinschreibung und Schrägstriche in der Befehlszeile

RaptorXML (und **RaptorXMLServer** für Administrator-Befehle) *unter Windows*
raptorxml und **raptorxmlserver** für Administrator-Befehle) *unter Windows und Unix (Linux, Mac)*

* Beachten Sie, dass klein geschriebene Befehle (raptorxml und raptorxmlserver) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (RaptorXML) nur unter Windows und Mac ausgeführt werden.
*Verwenden Sie auf Linux und Mac-Systemen Schrägstriche und auf Windows-Systemen umgekehrte Schrägstriche.

▼ Umgekehrte Schrägstriche, Leerzeichen und Sonderzeichen auf Windows-Systemen

Auf Windows-Systemen: Verwenden Sie bei Vorhandensein von Leerzeichen oder Sonerzeichen in Strings (z.B. in Datei- oder Ordnernamen oder Firmen-, Personen- oder Produktnamen) Anführungszeichen: z.B., **"Meine Datei"**. Beachten Sie jedoch, dass ein von einem doppelten Anführungszeichen gefolgter umgekehrter Schrägstrich (z.B: **"C:\Mein Verzeichnis\"**) eventuell nicht korrekt gelesen wird, da der umgekehrte Schrägstrich auch den Beginn einer Escape-Sequenz markiert und die Escape-Sequenz **\"** für ein doppeltes Anführungszeichen steht. Wenn Sie diese Zeichensequenz mit einem Escape versehen wollen, verwenden Sie einen vorangestellten umgekehrten Schrägstrich, wie den folgenden: **\\"**. Einfacher ausgedrückt: Wenn Sie einen Dateipfad, der Leerzeichen oder einen umgekehrten Schrägstrich am Ende enthält, schreiben müssen, so schreiben Sie diesen folgendermaßen: **"C:\Mein Verzeichnis\\"**.

Optionen

Optionen werden in ihrer kurzen Form (falls verfügbar) und in ihrer langen Form aufgelistet. Für die kurze und die lange Form können ein oder zwei Bindestriche verwendet werden. Eine Option kann, muss aber keinen Wert erhalten. Eine Option, die einen Wert erhält, wird folgendermaßen geschrieben: **--option=wert**. Werte können außer in zwei Fällen ohne Anführungszeichen definiert werden: (i) wenn der Wertestring Leerzeichen enthält oder (ii) wenn in der Beschreibung der Option explizit erwähnt ist, dass Anführungszeichen zwingend erforderlich sind. Wenn eine Option einen Booleschen Wert erhält und kein Wert definiert ist, so ist der Standardwert der Option TRUE. Mit Hilfe der Option **--h, --help** können Sie Informationen über den Befehl anzeigen.

▼ XQuery-Verarbeitung

▼ omit-xml-declaration

--omit-xml-declaration = true|false
Serialisierungsoption, mit der angegeben wird, ob die XML-Deklaration in der Ausgabe enthalten sein soll oder nicht. Wenn der Wert true ist, enthält das Ausgabedokument keine XML-Deklaration. Wenn der Wert false ist, wird eine XML-Deklaration inkludiert. Der Standardwert ist false.
Hinweis: Die Booleschen Optionswerte werden auf true gesetzt, wenn die Option ohne einen Wert definiert wird.

- xquery-version

  --xquery-version = `1|1.0|3|3.0|3.1`
  Gibt an, ob der XQuery-Prozessor XQuery 1.0 oder XQuery 3.0 verwenden soll. Der Standardwert ist 3.1.

- XML-Schema und XML-Instanz
  - load-xml-with-psvi

    --load-xml-with-psvi = `true|false`
    Ermöglicht die Validierung von XML-Input-Dateien und die Generierung von Informationen für diese Dateien nach Validierung des Schemas. Der Standardwert ist: `true`.

  - schema-imports

    --schema-imports = `load-by-schemalocation | load-preferring-schemalocation | load-by-namespace | load-combining-both | license-namespace-only`
    Definiert das Verhalten von `xs:import` Elementen, von denen jedes ein optionales `namespace` Attribut und ein optionales `schemaLocation` Attribut hat: `<import namespace="someNS" schemaLocation="someURL">`. Mit der Option wird definiert, ob ein Schema-Dokument geladen oder nur ein Namespace lizenziert werden soll und, wenn ein Schema-Dokument geladen werden soll, anhand welcher Informationen es gesucht werden soll. Standardeinstellung: `load-preferring-schemalocation`. Das Verhalten ist das folgende:

    - `load-by-schemalocation`: Anhand des Werts des `schemaLocation` Attributs wird der Schemapfad ermittelt, wobei Katalog-Mappings[48] berücksichtigt werden. Wenn das Namespace-Attribut vorhanden ist, wird der Namespace importiert (lizenziert).
    - `load-preferring-schemalocation`: Wenn das `schemaLocation` Attribut vorhanden ist, wird es verwendet, wobei Katalog-Mappings[48] berücksichtigt werden. Falls kein `schemaLocation` Attribut vorhanden ist, wird der Wert des `namespace` Attributs über ein Katalog-Mapping[48] verwendet. Dies ist der **Standardwert**.
    - `load-by-namespace`: Anhand des Werts des `namespace` Attributs wird der Schemapfad über ein Katalog-Mapping[48] ermittelt.
    - `load-combining-both`: Wenn entweder das Attribut `namespace` oder das Attribut `schemaLocation` ein Katalog-Mapping[48] hat, so wird das Mapping verwendet. Wenn beide Attribute Katalog-Mappings[48] haben, ist es vom Wert der Option `--schema-mapping` (XML/XSD-Option[255]) abhängig, welches Mapping verwendet wird. Falls kein Katalog-Mapping[48] vorhanden ist, wird das `schemaLocation` Attribut verwendet.
    - `license-namespace-only`: Der Namespace wird importiert. Kein Schema-Dokument wird importiert.

  - schema-location-hints

    --schema-location-hints = `load-by-schemalocation | load-by-namespace | load-combining-both | ignore`
    Definiert das Verhalten der Attribute `xsi:schemaLocation` und `xsi:noNamespaceSchemaLocation`: Legt fest, ob ein Schema-Dokument geladen werden soll und falls ja, anhand welcher Informationen es gesucht werden soll. Standardeinstellung: `load-by-schemalocation`.

    - Der Wert `load-by-schemalocation` verwendet die URL des Schemapfads[422] in den

Attributen `xsi:schemaLocation` und `xsi:noNamespaceSchemaLocation` in XML-Instanzdokumenten. Dies ist der **Standardwert**.

- Der Wert `load-by-namespace` verwendet den Namespace-Teil[422] von `xsi:schemaLocation` und - im Fall von `xsi:noNamespaceSchemaLocation` - einen leeren String und ermittelt das Schema über ein Katalog-Mapping[48].
- Bei Verwendung von `load-combining-both` und wenn entweder der Namespace-Teil oder der URL-Teil ein Katalog-Mapping[48] hat, so wird das Katalog-Mapping[48] verwendet. Wenn beide Katalog-Mappings[48] haben, ist es vom Wert der `--schema-mapping` Option (XML/XSD-Option[255]) abhängig, welches Mapping verwendet wird. Wenn weder der Namespace noch die URL ein Katalog-Mapping hat, wird die URL verwendet.
- Wenn der Wert der Option `ignore` ist, werden die beiden Attribute `xsi:schemaLocation` und `xsi:noNamespaceSchemaLocation` ignoriert.

▼ schema-mapping

`--schema-mapping =` `prefer-schemalocation | prefer-namespace`

Wenn ein Schema-Dokument sowohl anhand des Schema-Pfads als auch des Namespace gesucht werden soll, wird damit festgelegt, welche der beiden Optionen beim Katalog-Lookup Vorrang erhält. (Wenn eine der Optionen `--schemalocation-hints` oder `--schema-imports` einen Wert `load-combining-both` hat und wenn die betroffenen Namespace- und URL-Teile beide Katalog-Mappings[48] haben, gibt der Wert dieser Option an, welches der beiden Mappings verwendet werden soll (das Namespace Mapping oder das URL-Mapping; der Wert `prefer-schemalocation` bezieht sich auf das URL-Mapping)). Der Standardwert ist `prefer-schemalocation`.

▼ xinclude

`--xinclude =` `true|false`

Aktiviert die Unterstützung für XML-Inkludierungen (XInclude). Der Standardwert ist `false`. Bei `false` werden die `include`-Elemente von XInclude ignoriert.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

▼ xml-mode

`--xml-mode =` `wf|id|valid`

Definiert, welcher XML-Verarbeitungsmodus für das XML-Instanzdokument verwendet werden soll: `wf`=Wohlgeformtheitsprüfung; `id`=Wohlgeformtheitsprüfung mit ID/IDREF-Prüfung; `valid`=Validierung. Der Standardwert ist `wf`. Beachten Sie, dass jedes bei der Verarbeitung geladene Instanzdokument eine DTD referenzieren muss, damit der Wert `valid` ausgegeben werden kann. Falls keine DTD vorhanden ist, wird ein Fehler ausgegeben.

▼ xml-mode-for-schemas

`--xml-mode-for-schemas =` `wf|id|valid`

Definiert, welcher XML-Verarbeitungsmodus für XML-Schema-Dokumente verwendet werden soll: `wf`=Wohlgeformtheitsprüfung; `id`=Wohlgeformtheitsprüfung mit ID/IDREF-Prüfung; `valid`=Validierung. Der Standardwert ist `wf`. Beachten Sie, dass jedes bei der Verarbeitung geladene Schema-Dokument eine DTD referenzieren muss, damit der Wert `valid` ausgegeben werden kann. Falls keine DTD vorhanden ist, wird ein Fehler ausgegeben.

▼ xpath-static-type-errors-as-warnings

`--xpath-static-type-errors-as-warnings =` `true|false`

Bei `true` werden alle im statischen XPath-Kontext gefundenen Typ-Fehler auf eine Warnung herabgestuft. Während die Ausführung bei einem Fehler fehlschlagen würde, könnte sie bei einer

Warnung fortgesetzt werden. Die Standardeinstellung ist `false`.

- xsd-version

  `--xsd-version =` `1.0|1.1|detect`
  Definiert die zu verwendende W3C Schema Definition Language (XSD) Version. Der Standardwert ist `1.0`. Diese Option eignet sich auch, um herauszufinden, inwiefern ein 1.0-kompatibles Schema nicht mit Schemaversion 1.1 kompatibel ist. Die Option `detect` ist eine Altova-spezifische Funktionalität. Mit dieser Option kann die Version des XML-Schema-Dokuments (`1.0` oder `1.1`) durch Lesen des Werts des `vc:minVersion` Attributs des `<xs:schema>` Elements des Dokuments ermittelt werden. Wenn der Wert des `@vc:minVersion` Attributs `1.1` ist, wird das Schema als Version `1.1` erkannt. Bei jedem anderen Wert wird das Schema als `1.0` erkannt. Bei jedem anderen Wert oder bei Fehlen des `@vc:minVersion` Attributs wird das Schema als Version `1.0` gelesen.

Kataloge und globale Ressourcen

- catalog

  `--catalog =` *`FILE`*
  Gibt den absoluten Pfad zu einer Root-Katalog-Datei an, die nicht die installierte Root-Katalog-Datei ist. Der Standardwert ist der absolute Pfad zur installierten Root-Katalog-Datei. (`<installationsordner>\Altova\RaptorXMLServer2025\etc\RootCatalog.xml`). Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge[48].

- user-catalog

  `--user-catalog =` *`FILE`*
  Definiert den absoluten Pfad zu einem XML-Katalog, der zusätzlich zum Root-Katalog verwendet werden soll. Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge[48].

- enable-globalresources

  `--enable-globalresources =` `true|false`
  Aktiviert die globalen Ressourcen[55]. Standardwert ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- globalresourceconfig [gc]

  `--gc | --globalresourceconfig =` *`VALUE`*
  Definiert die aktive Konfiguration der globalen Ressource[55] (und aktiviert globale Ressourcen)[55].

- globalresourcefile [gr]

  `--gr | --globalresourcefile =` *`FILE`*
  Definiert die globale Ressourcendatei[55] (und aktiviert globale Ressourcen)[55].

Erweiterungen

Diese Optionen definieren die Behandlung von speziellen Erweiterungsfunktionen, die in einer Reihe von Enterprise Versionen von Altova-Produkten (wie z.B. in XMLSpy Enterprise Edition) verfügbar sind. Die Verwendung dieser Funktionen ist im Benutzerhandbuch des jeweiligen Produkts beschrieben.

- chartext-disable

  --chartext-disable = `true|false`
  Deaktiviert Diagrammerweiterungen. Der Standardwert ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- dotnetext-disable

  --dotnetext-disable = `true|false`
  Deaktiviert .NET-Erweiterungen. Der Standardwert ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- jvm-location

  --jvm-location = *FILE*
  FILE definiert den Pfad zur Java Virtual Machine (DLL unter Windows, freigegebenes Objekt unter Linux). Sie benötigen JVM, wenn Sie Java-Erweiterungsfunktionen 530 in Ihrem XSLT/XQuery-Code verwenden. Die Standardeinstellung ist `false`.

- javaext-barcode-location

  --javaext-barcode-location = *FILE*
  Definiert den Pfad zum Ordner, der die Barcode-Erweiterungsdatei `AltovaBarcodeExtension.jar` enthält. Der Pfad muss in einer der folgenden Formen angegeben werden:

  - als Datei-URI, z.B.: `--javaext-barcode-location="file:///C:/Program Files/Altova/RaptorXMLServer2025/etc/jar/"`
  - als Windows-Pfad mit maskierten umgekehrten Schrägstrichen, z.B.: `--javaext-barcode-location="C:\\Program Files\\Altova\\RaptorXMLServer2025\\etc\\jar\\"`

- javaext-disable

  --javaext-disable = `true|false`
  Deaktiviert Java-Erweiterungen. Der Standardwert ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- Allgemeine Optionen

  - error-format

    --error-format = `text|shortxml|longxml`
    Definiert das Format der Fehlerausgabe. Der Standardwert ist `text`. Mit den anderen Optionen werden XML-Formate generiert, wobei mit `longxml` mehr Details generiert werden.

  - error-limit

    --error-limit = *N | unlimited*
    Definiert das Fehlerlimit mit einem Wertebereich von `1` bis `9999` oder `unbegrenzt`. Der Standardwert ist `100`. Bei Erreichung des Fehlerlimits wird die Validierung gestoppt. Dient dazu, die Prozessorverwendung während der Validierung/Transformation einzuschränken.

- info-limit

  --info-limit = *N | unlimited*
  Definiert die Grenze für Informationsmeldungen im Bereich von `1-65535` oder als `unlimited` (unbegrenzt). Die Verarbeitung wird auch nach Erreichen des Info-Limits fortgesetzt, doch werden weitere Meldungen nicht mehr ausgegeben. Der Standardwert ist `100`.

- help

  --help
  Zeigt den Hilfetext zum Befehl an. Beispiel: `valany --h`. (Alternativ dazu kann der Befehl `help` zusammen mit einem Argument verwendet werden. Beispiel: `help valany`.)

- listfile

  --listfile = `true|false`
  Bei `true` wird das Argument *InputFile* des Befehls als Textdatei behandelt, die einen Dateinamen pro Zeile enthält. Der Standardwert ist `false`. (Als Alternative können die Dateien im CLI getrennt durch ein Leerzeichen aufgelistet werden. Beachten Sie allerdings, dass CLIs eine maximale Zeichenanzahl haben). Beachten Sie, dass die Option `--listfile` nur auf Argumente, nicht aber auf Optionen angewendet wird.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- log-output

  --log-output = *FILE*
  Schreibt die Meldungsausgabe in die angegebene URL. Stellen Sie sicher, dass das CLI Schreibrechte für den Ausgabepfad hat.

- network-timeout

  --network-timeout = *VALUE*
  Definiert das Timeout für entfernte I/O-Operationen in Millisekunden. Der Standardwert ist: `40000`.

- recurse

  --recurse = `true|false`
  Dient zur Auswahl von Dateien innerhalb von Unterverzeichnissen einschließlich ZIP-Archiven. Bei `true` wählt das Argument *InputFile* des Befehls die angegebene Datei auch in den Unterverzeichnissen aus. Beispiel: `"test.zip|zip\test.xml"` wählt Dateien mit dem Namen `test.xml` auf allen Ordnerebenen des ZIP-Ordners aus. Referenzen auf ZIP-Dateien müssen in Anführungszeichen angegeben werden. Es können die Platzhalter `*` und `?` verwendet werden. Mit `*.xml` werden folglich alle `.xml` Dateien im (ZIP-)Ordner ausgewählt. Der Standardwert der Option ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- verbose

  --verbose = `true|false`
  Mit dem Wert `true` wird die Ausgabe zusätzlicher Informationen bei der Validierung aktiviert. Der

Standardwert ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- verbose-output

  `--verbose-output =` *`FILE`*
  Schreibt die ausführliche Ausgabe in *`FILE`*.

- version

  `--version`
  Zeigt die Version von RaptorXML Server an. Setzen Sie `--version` bei Verwendung mit einem Befehl vor den Befehl.

- warning-limit

  `--warning-limit =` *`N | unlimited`*
  Definiert das Warnungslimit im Bereich von `1-65535` oder `unbegrenzt`. Bei Erreichen dieses Limits wird die Verarbeitung fortgesetzt, doch werden keine weiteren Warnungen mehr ausgegeben. Der Standardwert ist `100`.

5.3.4 valxqueryupdate

Der Befehl **`valxqueryupdate`** erhält als einziges Argument eine XQuery-Datei und validiert diese.

```
raptorxml valxqueryupdate [options] XQuery-File
```

- Das Argument ***`XQuery-File`*** ist der Pfad und Name der zu validierenden XQuery-Datei.

Beispiele

Beispiele für den Befehl **`valxqueryupdate`**:

- **`raptorxml`** `valxqueryupdate c:\Test.xqu`
- **`raptorxml`** `valxqueryupdate --xquery-update-version=1 c:\Test.xqu`

- Groß- und Kleinschreibung und Schrägstriche in der Befehlszeile

  `RaptorXML` (und **`RaptorXMLServer`** für Administrator-Befehle) *unter Windows*
  `raptorxml` und **`raptorxmlserver`** für Administrator-Befehle) *unter Windows und Unix (Linux, Mac)*

  * Beachten Sie, dass klein geschriebene Befehle (`raptorxml` und `raptorxmlserver`) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (`RaptorXML`) nur unter Windows und Mac ausgeführt werden.
  *Verwenden Sie auf Linux und Mac-Systemen Schrägstriche und auf Windows-Systemen umgekehrte Schrägstriche.

- Umgekehrte Schrägstriche, Leerzeichen und Sonderzeichen auf Windows-Systemen

  Auf Windows-Systemen: Verwenden Sie bei Vorhandensein von Leerzeichen oder Sonerzeichen in Strings (z.B. in Datei- oder Ordnernamen oder Firmen-, Personen- oder Produktnamen) Anführungszeichen: z.B., **"Meine Datei"**. Beachten Sie jedoch, dass ein von einem doppelten Anführungszeichen gefolgter umgekehrter Schrägstrich (z.B: **"C:\Mein Verzeichnis\"**) eventuell nicht korrekt gelesen wird, da der umgekehrte Schrägstrich auch den Beginn einer Escape-Sequenz markiert und die Escape-Sequenz **\"** für ein doppeltes Anführungszeichen steht. Wenn Sie diese Zeichensequenz mit einem Escape versehen wollen, verwenden Sie einen vorangestellten umgekehrten Schrägstrich, wie den folgenden: **\\"**. Einfacher ausgedrückt: Wenn Sie einen Dateipfad, der Leerzeichen oder einen umgekehrten Schrägstrich am Ende enthält, schreiben müssen, so schreiben Sie diesen folgendermaßen: **"C:\Mein Verzeichnis\\"**.

Optionen

Optionen werden in ihrer kurzen Form (falls verfügbar) und in ihrer langen Form aufgelistet. Für die kurze und die lange Form können ein oder zwei Bindestriche verwendet werden. Eine Option kann, muss aber keinen Wert erhalten. Eine Option, die einen Wert erhält, wird folgendermaßen geschrieben: **--option=wert**. Werte können außer in zwei Fällen ohne Anführungszeichen definiert werden: (i) wenn der Wertestring Leerzeichen enthält oder (ii) wenn in der Beschreibung der Option explizit erwähnt ist, dass Anführungszeichen zwingend erforderlich sind. Wenn eine Option einen Booleschen Wert erhält und kein Wert definiert ist, so ist der Standardwert der Option `TRUE`. Mit Hilfe der Option **--h, --help** können Sie Informationen über den Befehl anzeigen.

- XQuery-Verarbeitung
  - omit-xml-declaration

    --omit-xml-declaration = `true|false`
    Serialisierungsoption, mit der angegeben wird, ob die XML-Deklaration in der Ausgabe enthalten sein soll oder nicht. Wenn der Wert `true` ist, enthält das Ausgabedokument keine XML-Deklaration. Wenn der Wert `false` ist, wird eine XML-Deklaration inkludiert. Der Standardwert ist `false`.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

  - xquery-update-version

    --xquery-update-version = `1|1.0|3|3.0|`
    Definiert, ob der XQuery-Prozessor XQuery Update Facility 1.0 oder XQuery Update Facility 3.0 verwenden soll. Der Standardwert ist `3`.

- XML-Schema und XML-Instanz
  - load-xml-with-psvi

    --load-xml-with-psvi = `true|false`
    Ermöglicht die Validierung von XML-Input-Dateien und die Generierung von Informationen für diese Dateien nach Validierung des Schemas. Der Standardwert ist: `true`.

  - schema-imports

    --schema-imports = `load-by-schemalocation | load-preferring-schemalocation | load-`

`by-namespace | load-combining-both | license-namespace-only`
Definiert das Verhalten von `xs:import` Elementen, von denen jedes ein optionales `namespace` Attribut und ein optionales `schemaLocation` Attribut hat: `<import namespace="someNS" schemaLocation="someURL">`. Mit der Option wird definiert, ob ein Schema-Dokument geladen oder nur ein Namespace lizenziert werden soll und, wenn ein Schema-Dokument geladen werden soll, anhand welcher Informationen es gesucht werden soll. Standardeinstellung: `load-preferring-schemalocation`. Das Verhalten ist das folgende:

- `load-by-schemalocation`: Anhand des Werts des `schemaLocation` Attributs wird der Schemapfad ermittelt, wobei Katalog-Mappings[48] berücksichtigt werden. Wenn das Namespace-Attribut vorhanden ist, wird der Namespace importiert (lizenziert).
- `load-preferring-schemalocation`: Wenn das `schemaLocation` Attribut vorhanden ist, wird es verwendet, wobei Katalog-Mappings[48] berücksichtigt werden. Falls kein `schemaLocation` Attribut vorhanden ist, wird der Wert des `namespace` Attributs über ein Katalog-Mapping[48] verwendet. Dies ist der **Standardwert**.
- `load-by-namespace`: Anhand des Werts des `namespace` Attributs wird der Schemapfad über ein Katalog-Mapping[48] ermittelt.
- `load-combining-both`: Wenn entweder das Attribut `namespace` oder das Attribut `schemaLocation` ein Katalog-Mapping[48] hat, so wird das Mapping verwendet. Wenn beide Attribute Katalog-Mappings[48] haben, ist es vom Wert der Option `--schema-mapping` (XML/XSD-Option[255]) abhängig, welches Mapping verwendet wird. Falls kein Katalog-Mapping[48] vorhanden ist, wird das `schemaLocation` Attribut verwendet.
- `license-namespace-only`: Der Namespace wird importiert. Kein Schema-Dokument wird importiert.

▼ schema-location-hints

`--schema-location-hints =` `load-by-schemalocation | load-by-namespace | load-combining-both | ignore`
Definiert das Verhalten der Attribute `xsi:schemaLocation` und `xsi:noNamespaceSchemaLocation`: Legt fest, ob ein Schema-Dokument geladen werden soll und falls ja, anhand welcher Informationen es gesucht werden soll. Standardeinstellung: `load-by-schemalocation`.

- Der Wert `load-by-schemalocation` verwendet die URL des Schemapfads[422] in den Attributen `xsi:schemaLocation` und `xsi:noNamespaceSchemaLocation` in XML-Instanzdokumenten. Dies ist der **Standardwert**.
- Der Wert `load-by-namespace` verwendet den Namespace-Teil[422] von `xsi:schemaLocation` und - im Fall von `xsi:noNamespaceSchemaLocation` - einen leeren String und ermittelt das Schema über ein Katalog-Mapping[48].
- Bei Verwendung von `load-combining-both` und wenn entweder der Namespace-Teil oder der URL-Teil ein Katalog-Mapping[48] hat, so wird das Katalog-Mapping[48] verwendet. Wenn beide Katalog-Mappings[48] haben, ist es vom Wert der `--schema-mapping` Option (XML/XSD-Option[255]) abhängig, welches Mapping verwendet wird. Wenn weder der Namespace noch die URL ein Katalog-Mapping hat, wird die URL verwendet.
- Wenn der Wert der Option `ignore` ist, werden die beiden Attribute `xsi:schemaLocation` und `xsi:noNamespaceSchemaLocation` ignoriert.

▼ schema-mapping

`--schema-mapping =` `prefer-schemalocation | prefer-namespace`
Wenn ein Schema-Dokument sowohl anhand des Schema-Pfads als auch des Namespace gesucht werden soll, wird damit festgelegt, welche der beiden Optionen beim Katalog-Lookup Vorrang erhält. (Wenn eine der Optionen `--schemalocation-hints` oder `--schema-imports` einen Wert `load-`

`combining-both` hat und wenn die betroffenen Namespace- und URL-Teile beide Katalog-Mappings[48] haben, gibt der Wert dieser Option an, welches der beiden Mappings verwendet werden soll (das Namespace Mapping oder das URL-Mapping; der Wert `prefer-schemalocation` bezieht sich auf das URL-Mapping)). Der Standardwert ist `prefer-schemalocation`.

- xinclude

  `--xinclude =` `true|false`
  Aktiviert die Unterstützung für XML-Inkludierungen (XInclude). Der Standardwert ist `false`. Bei `false` werden die `include`-Elemente von XInclude ignoriert.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- xml-mode

  `--xml-mode =` `wf|id|valid`
  Definiert, welcher XML-Verarbeitungsmodus für das XML-Instanzdokument verwendet werden soll: `wf`=Wohlgeformtheitsprüfung; `id`=Wohlgeformtheitsprüfung mit ID/IDREF-Prüfung; `valid`=Validierung. Der Standardwert ist `wf`. Beachten Sie, dass jedes bei der Verarbeitung geladene Instanzdokument eine DTD referenzieren muss, damit der Wert `valid` ausgegeben werden kann. Falls keine DTD vorhanden ist, wird ein Fehler ausgegeben.

- xml-mode-for-schemas

  `--xml-mode-for-schemas =` `wf|id|valid`
  Definiert, welcher XML-Verarbeitungsmodus für XML-Schema-Dokumente verwendet werden soll: `wf`=Wohlgeformtheitsprüfung; `id`=Wohlgeformtheitsprüfung mit ID/IDREF-Prüfung; `valid`=Validierung. Der Standardwert ist `wf`. Beachten Sie, dass jedes bei der Verarbeitung geladene Schema-Dokument eine DTD referenzieren muss, damit der Wert `valid` ausgegeben werden kann. Falls keine DTD vorhanden ist, wird ein Fehler ausgegeben.

- xpath-static-type-errors-as-warnings

  `--xpath-static-type-errors-as-warnings =` `true|false`
  Bei `true` werden alle im statischen XPath-Kontext gefundenen Typ-Fehler auf eine Warnung herabgestuft. Während die Ausführung bei einem Fehler fehlschlagen würde, könnte sie bei einer Warnung fortgesetzt werden. Die Standardeinstellung ist `false`.

- xsd-version

  `--xsd-version =` `1.0|1.1|detect`
  Definiert die zu verwendende W3C Schema Definition Language (XSD) Version. Der Standardwert ist `1.0`. Diese Option eignet sich auch, um herauszufinden, inwiefern ein 1.0-kompatibles Schema nicht mit Schemaversion 1.1 kompatibel ist. Die Option `detect` ist eine Altova-spezifische Funktionalität. Mit dieser Option kann die Version des XML-Schema-Dokuments (`1.0` oder `1.1`) durch Lesen des Werts des `vc:minVersion` Attributs des `<xs:schema>` Elements des Dokuments ermittelt werden. Wenn der Wert des `@vc:minVersion` Attributs `1.1` ist, wird das Schema als Version `1.1` erkannt. Bei jedem anderen Wert wird das Schema als `1.0` erkannt. Bei jedem anderen Wert oder bei Fehlen des `@vc:minVersion` Attributs wird das Schema als Version `1.0` gelesen.

- Kataloge und globale Ressourcen
  - catalog

--catalog = *FILE*
Gibt den absoluten Pfad zu einer Root-Katalog-Datei an, die nicht die installierte Root-Katalog-Datei ist. Der Standardwert ist der absolute Pfad zur installierten Root-Katalog-Datei. (`<installationsordner>\Altova\RaptorXMLServer2025\etc\RootCatalog.xml`). Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge[48].

- user-catalog

  --user-catalog = *FILE*
  Definiert den absoluten Pfad zu einem XML-Katalog, der zusätzlich zum Root-Katalog verwendet werden soll. Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge[48].

- enable-globalresources

  --enable-globalresources = `true|false`
  Aktiviert die globalen Ressourcen[55]. Standardwert ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- globalresourceconfig [gc]

  --gc | --globalresourceconfig = *VALUE*
  Definiert die aktive Konfiguration der globalen Ressource[55] (und aktiviert globale Ressourcen)[55].

- globalresourcefile [gr]

  --gr | --globalresourcefile = *FILE*
  Definiert die globale Ressourcendatei[55] (und aktiviert globale Ressourcen)[55].

- Erweiterungen

  Diese Optionen definieren die Behandlung von speziellen Erweiterungsfunktionen, die in einer Reihe von Enterprise Versionen von Altova-Produkten (wie z.B. in XMLSpy Enterprise Edition) verfügbar sind. Die Verwendung dieser Funktionen ist im Benutzerhandbuch des jeweiligen Produkts beschrieben.

  - chartext-disable

    --chartext-disable = `true|false`
    Deaktiviert Diagrammerweiterungen. Der Standardwert ist `false`.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

  - dotnetext-disable

    --dotnetext-disable = `true|false`
    Deaktiviert .NET-Erweiterungen. Der Standardwert ist `false`.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

  - jvm-location

    --jvm-location = *FILE*
    FILE definiert den Pfad zur Java Virtual Machine (DLL unter Windows, freigegebenes Objekt unter Linux). Sie benötigen JVM, wenn Sie Java-Erweiterungsfunktionen[530] in Ihrem XSLT/XQuery-Code

verwenden. Die Standardeinstellung ist `false`.

- javaext-barcode-location

  `--javaext-barcode-location =` *`FILE`*
  Definiert den Pfad zum Ordner, der die Barcode-Erweiterungsdatei `AltovaBarcodeExtension.jar` enthält. Der Pfad muss in einer der folgenden Formen angegeben werden:

  - als Datei-URI, z.B.: `--javaext-barcode-location="file:///C:/Program Files/Altova/RaptorXMLServer2025/etc/jar/"`
  - als Windows-Pfad mit maskierten umgekehrten Schrägstrichen, z.B.: `--javaext-barcode-location="C:\\Program Files\\Altova\\RaptorXMLServer2025\\etc\\jar\\"`

- javaext-disable

  `--javaext-disable =` `true|false`
  Deaktiviert Java-Erweiterungen. Der Standardwert ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- Allgemeine Optionen

  - error-format

    `--error-format =` `text|shortxml|longxml`
    Definiert das Format der Fehlerausgabe. Der Standardwert ist `text`. Mit den anderen Optionen werden XML-Formate generiert, wobei mit `longxml` mehr Details generiert werden.

  - error-limit

    `--error-limit =` *`N | unlimited`*
    Definiert das Fehlerlimit mit einem Wertebereich von `1` bis `9999` oder `unbegrenzt`. Der Standardwert ist `100`. Bei Erreichung des Fehlerlimits wird die Validierung gestoppt. Dient dazu, die Prozessorverwendung während der Validierung/Transformation einzuschränken.

  - info-limit

    `--info-limit =` *`N | unlimited`*
    Definiert die Grenze für Informationsmeldungen im Bereich von `1-65535` oder als `unlimited` (unbegrenzt). Die Verarbeitung wird auch nach Erreichen des Info-Limits fortgesetzt, doch werden weitere Meldungen nicht mehr ausgegeben. Der Standardwert ist `100`.

  - help

    `--help`
    Zeigt den Hilfetext zum Befehl an. Beispiel: `valany --h`. (Alternativ dazu kann der Befehl `help` zusammen mit einem Argument verwendet werden. Beispiel: `help valany`.)

  - listfile

    `--listfile =` `true|false`
    Bei `true` wird das Argument *`InputFile`* des Befehls als Textdatei behandelt, die einen Dateinamen pro Zeile enthält. Der Standardwert ist `false`. (Als Alternative können die Dateien im CLI getrennt

durch ein Leerzeichen aufgelistet werden. Beachten Sie allerdings, dass CLIs eine maximale Zeichenanzahl haben). Beachten Sie, dass die Option `--listfile` nur auf Argumente, nicht aber auf Optionen angewendet wird.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- log-output

  `--log-output =` *`FILE`*
  Schreibt die Meldungsausgabe in die angegebene URL. Stellen Sie sicher, dass das CLI Schreibrechte für den Ausgabepfad hat.

- network-timeout

  `--network-timeout =` *`VALUE`*
  Definiert das Timeout für entfernte I/O-Operationen in Millisekunden. Der Standardwert ist: `40000`.

- recurse

  `--recurse =` `true|false`
  Dient zur Auswahl von Dateien innerhalb von Unterverzeichnissen einschließlich ZIP-Archiven. Bei `true` wählt das Argument *`InputFile`* des Befehls die angegebene Datei auch in den Unterverzeichnissen aus. Beispiel: `"test.zip|zip\test.xml"` wählt Dateien mit dem Namen `test.xml` auf allen Ordnerebenen des ZIP-Ordners aus. Referenzen auf ZIP-Dateien müssen in Anführungszeichen angegeben werden. Es können die Platzhalter `*` und `?` verwendet werden. Mit `*.xml` werden folglich alle `.xml` Dateien im (ZIP-)Ordner ausgewählt. Der Standardwert der Option ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- verbose

  `--verbose =` `true|false`
  Mit dem Wert `true` wird die Ausgabe zusätzlicher Informationen bei der Validierung aktiviert. Der Standardwert ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- verbose-output

  `--verbose-output =` *`FILE`*
  Schreibt die ausführliche Ausgabe in *`FILE`*.

- version

  `--version`
  Zeigt die Version von RaptorXML Server an. Setzen Sie `--version` bei Verwendung mit einem Befehl vor den Befehl.

- warning-limit

  `--warning-limit =` *`N | unlimited`*
  Definiert das Warnungslimit im Bereich von `1-65535` oder `unbegrenzt`. Bei Erreichen dieses Limits

wird die Verarbeitung fortgesetzt, doch werden keine weiteren Warnungen mehr ausgegeben. Der Standardwert ist `100`.

5.4 XSLT-Befehle

Die XSLT-Befehle sind:

- xslt 130: zum Transformieren von XML-Dokumenten anhand eines XSLT-Dokuments
- valxslt 139: zum Validieren von XSLT-Dokumenten

5.4.1 xslt

Der Befehl **xslt** erhält als einziges Argument eine XSLT-Datei und transformiert anhand dieser Datei eine XML-Input-Datei in eine Ausgabedatei. Die Input-Datei und die Ausgabedatei sind als Optionen 259 definiert.

```
raptorxml xslt [options] XSLT-File
```

- Das Argument *XSLT-File* ist der Pfad und Name der für die Transformation zu verwendenden XSLT-Datei.
- Es wird eine XML-Input-Datei (--input 259) oder eine benannte Vorlage als Eintrittspunkt (--template-entry-point 259) benötigt.
- Um JSON-Daten zu transformieren, laden Sie die JSON-Daten über die XPath 3.1-Funktion **json-doc($path)** und verwenden Sie die **--initial-match-selection**-Option des **xslt**-Befehls. Siehe letzter Beispielpunkt in den Bespielen unten.
- Wenn keine --output 259 Option definiert ist, wird die Ausgabe in die Standardausgabe geschrieben. Sie können XSLT 1.0, 2.0 oder 3.0 verwenden. Standardmäßig wird XSLT 3.0 verwendet.

Beispiele

Beispiele für den Befehl **xslt**:

- **raptorxml** xslt --input=c:\Test.xml --output=c:\Output.xml c:\Test.xslt
- **raptorxml** xslt --template-entry-point=StartTemplate --output=c:\Output.xml c:\Test.xslt
- **raptorxml** xslt --input=c:\Test.xml --output=c:\Output.xml --param=date://node[1]/@att1 --p=title:'stringwithoutspace' --param=title:"'string with spaces'" --p=amount:456 c:\Test.xslt
- **raptorxml** xslt --initial-match-selection=json-doc('MyData.json', map{'liberal':true()}) --output=c:\MyData.xml c:\Test.xslt
- **raptorxml** xslt --initial-match-selection="**json-doc(**'MyData.json', **map**{'liberal':true()}**)**" --output=c:\MyData.xml c:\Test.xslt (If the json-doc argument string contains spaces, then enclose the entire json-doc value in quotes.)

▼ Groß- und Kleinschreibung und Schrägstriche in der Befehlszeile

RaptorXML (und **RaptorXMLServer** für Administrator-Befehle) *unter Windows*
raptorxml und **raptorxmlserver** für Administrator-Befehle) *unter Windows und Unix (Linux, Mac)*

* Beachten Sie, dass klein geschriebene Befehle (raptorxml und raptorxmlserver) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (RaptorXML) nur unter Windows und Mac ausgeführt werden.
*Verwenden Sie auf Linux und Mac-Systemen Schrägstriche und auf Windows-Systemen umgekehrte

Schrägstriche.

- Umgekehrte Schrägstriche, Leerzeichen und Sonderzeichen auf Windows-Systemen

  Auf Windows-Systemen: Verwenden Sie bei Vorhandensein von Leerzeichen oder Sonerzeichen in Strings (z.B. in Datei- oder Ordnernamen oder Firmen-, Personen- oder Produktnamen) Anführungszeichen: z.B., `"Meine Datei"`. Beachten Sie jedoch, dass ein von einem doppelten Anführungszeichen gefolgter umgekehrter Schrägstrich (z.B: `"C:\Mein Verzeichnis\"`) eventuell nicht korrekt gelesen wird, da der umgekehrte Schrägstrich auch den Beginn einer Escape-Sequenz markiert und die Escape-Sequenz `\"` für ein doppeltes Anführungszeichen steht. Wenn Sie diese Zeichensequenz mit einem Escape versehen wollen, verwenden Sie einen vorangestellten umgekehrten Schrägstrich, wie den folgenden: `\\"`. Einfacher ausgedrückt: Wenn Sie einen Dateipfad, der Leerzeichen oder einen umgekehrten Schrägstrich am Ende enthält, schreiben müssen, so schreiben Sie diesen folgendermaßen: `"C:\Mein Verzeichnis\\"`.

Optionen

Optionen werden in ihrer kurzen Form (falls verfügbar) und in ihrer langen Form aufgelistet. Für die kurze und die lange Form können ein oder zwei Bindestriche verwendet werden. Eine Option kann, muss aber keinen Wert erhalten. Eine Option, die einen Wert erhält, wird folgendermaßen geschrieben: `--option=wert`. Werte können außer in zwei Fällen ohne Anführungszeichen definiert werden: (i) wenn der Wertestring Leerzeichen enthält oder (ii) wenn in der Beschreibung der Option explizit erwähnt ist, dass Anführungszeichen zwingend erforderlich sind. Wenn eine Option einen Booleschen Wert erhält und kein Wert definiert ist, so ist der Standardwert der Option `TRUE`. Mit Hilfe der Option `--h, --help` können Sie Informationen über den Befehl anzeigen.

- XSLT-Verarbeitung
  - indent-characters

    `--indent-characters =` *VALUE*
    Definiert den Zeichenstring, der als Einrückung verwendet werden soll.

  - function-param

    `--function-param =` *VALUE*
    Definiert die Funktionen, die an die Anfangsfunktion übergeben werden. Um mehr als eine Funktion zu definieren, verwenden Sie die Option mehrmals. Beachten Sie jedoch, dass die Reihenfolge eine Rolle spielt.

  - global-context-item

    `--global-context-item =` *VALUE*
    Definiert das Kontextelement, das zur Evaluierung der globalen Variablen verwendet werden soll.

  - initial-function

    `--initial-function =` *VALUE*
    Der Name einer Funktion, die als Eintrittspunkt der Transformation ausgeführt werden soll.

  - initial-match-selection

    `--initial-match-selection =` *VALUE*
    Definiert den Wert (Reihenfolge) der Auswahl für die erste Übereinstimmung.

- initial-mode, template-mode

  --initial-mode, --template-mode = *VALUE*
  Definiert den Vorlagenmodus für die Transformation.

- initial-template, template-entry-point

  --initial-template, --template-entry-point = *VALUE*
  Gibt den Namen einer benannten Vorlage im XSLT-Stylesheet an, das der Eintrittspunkt der Transformation ist.

- input

  --input = *FILE*
  Die URL der zu transformierenden XML-Datei.

- output, xsltoutput

  output = *FILE*, **xsltoutput =** *FILE*
  Die URL der primären Ausgabedatei. So ist z.B. im Fall der Ausgabe mehrerer HTML-Dateien die primäre Ausgabedatei der Pfad der Eintrittspunkt-HTML-Datei. Zusätzliche Ausgabedateien wie z.B. generierte Bilddateien werden als `xslt-additional-output-files` angegeben. Wenn keine `--output` oder `--xsltoutput` Option definiert ist, wird die Ausgabe in die Standardausgabe geschrieben.

- param [p]

  --p | --param = *KEY:VALUE*

  XQuery

  Definiert den Wert eines externen Parameters. Ein externer Parameter ist im XQuery-Dokument mit der `declare variable` Deklaration gefolgt von einem Variablennamen und anschließend dem Schlüsselwort `external`, gefolgt von einem Semikolon deklariert. Beispiel:

  ```
  declare variable $foo as xs:string external;
  ```

  Aufgrund des Schlüsselworts `external` wird `$foo` zu einem externen Parameter, dessen Wert zur Laufzeit von einer externen Quelle aus übergeben wird. Der externe Parameter erhält mit dem CLI-Befehl einen Wert. Beispiel:

  ```
  --param=foo:'MyName'
  ```

  In der obigen Beschreibungsanweisung ist *KEY* der Name des externen Parameters, *VALUE* der als XPath-Ausdruck angegebene Wert des externen Parameters. Im CLI verwendete Parameter müssen im XQuery-Dokument deklariert sein. Wenn mehrere externe Parameter als Werte an das CLI übergeben werden, muss jeder eine separate `--param` Option erhalten. Wenn der XPath-Ausdruck Leerzeichen enthält, muss er in doppelte Anführungszeichen gesetzt werden.

  XSLT

  Definiert einen globalen Stylesheet-Parameter. *KEY* ist der Parametername, *VALUE* der als XPath-Ausdruck angegebene Parameterwert. Im CLI verwendete Parameter müssen im Stylesheet deklariert sein. Wenn mehrere Parameter verwendet werden, muss vor jedem Parameter die `--param` Option verwendet werden. Wenn der XPath-Ausdruck Leerzeichen

enthält - ob im XPath-Ausdruck selbst oder in einem String-Literal im Ausdruck - muss er in doppelte Anführungszeichen gesetzt werden. Beispiel:

```
raptorxml xslt --input=c:\Test.xml --output=c:\Output.xml --
param=date://node[1]/@att1 --p=title:'stringwithoutspace' --
param=title:"'string with spaces'" --p=amount:456 c:\Test.xslt
```

- streaming-serialization-enabled

  --streaming-serialization-enabled = `true|false`
  Aktiviert die Streaming-Serialisierung. Der Standardwert ist `true`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- template-param

  --template-param = *KEY:VALUE*
  Definiert die Parameter, die nur an die Anfangsvorlage (und nicht an untergeordnete Vorlagenaufrufe) übergeben werden sollen. Um mehrere Parameter zu definieren, verwenden Sie die Option für jeden Parameter je einmal.

- tunnel-param

  --tunnel-param = *KEY:VALUE*
  Definiert die Parameter, die an die Anfangsvorlage und an untergeordnete Vorlagenaufrufe übergeben werden sollen. Um mehrere Parameter zu definieren, verwenden Sie die Option für jeden Parameter je einmal.

- xpath-static-type-errors-as-warnings

  --xpath-static-type-errors-as-warnings = `true|false`
  Bei `true` werden alle im statischen XPath-Kontext gefundenen Typ-Fehler auf eine Warnung herabgestuft. Während die Ausführung bei einem Fehler fehlschlagen würde, könnte sie bei einer Warnung fortgesetzt werden. Die Standardeinstellung ist `false`.

- xslt-version

  --xslt-version = `1|1.0|2|2.0|3|3.0|3.1`
  Definiert, ob der XSLT-Prozessor XSLT 1.0, XSLT 2.0 oder XSLT 3.0 verwenden soll. Der Standardwert ist `3`.

- XML-Schema und XML-Instanz
  - load-xml-with-psvi

    --load-xml-with-psvi = `true|false`
    Ermöglicht die Validierung von XML-Input-Dateien und die Generierung von Informationen für diese Dateien nach Validierung des Schemas. Der Standardwert ist: `true`.

  - schema-imports

    --schema-imports = `load-by-schemalocation | load-preferring-schemalocation | load-`

`by-namespace | load-combining-both | license-namespace-only`
Definiert das Verhalten von `xs:import` Elementen, von denen jedes ein optionales `namespace` Attribut und ein optionales `schemaLocation` Attribut hat: `<import namespace="someNS" schemaLocation="someURL">`. Mit der Option wird definiert, ob ein Schema-Dokument geladen oder nur ein Namespace lizenziert werden soll und, wenn ein Schema-Dokument geladen werden soll, anhand welcher Informationen es gesucht werden soll. Standardeinstellung: `load-preferring-schemalocation`. Das Verhalten ist das folgende:

- `load-by-schemalocation`: Anhand des Werts des `schemaLocation` Attributs wird der Schemapfad ermittelt, wobei Katalog-Mappings[48] berücksichtigt werden. Wenn das Namespace-Attribut vorhanden ist, wird der Namespace importiert (lizenziert).
- `load-preferring-schemalocation`: Wenn das `schemaLocation` Attribut vorhanden ist, wird es verwendet, wobei Katalog-Mappings[48] berücksichtigt werden. Falls kein `schemaLocation` Attribut vorhanden ist, wird der Wert des `namespace` Attributs über ein Katalog-Mapping[48] verwendet. Dies ist der **Standardwert**.
- `load-by-namespace`: Anhand des Werts des `namespace` Attributs wird der Schemapfad über ein Katalog-Mapping[48] ermittelt.
- `load-combining-both`: Wenn entweder das Attribut `namespace` oder das Attribut `schemaLocation` ein Katalog-Mapping[48] hat, so wird das Mapping verwendet. Wenn beide Attribute Katalog-Mappings[48] haben, ist es vom Wert der Option `--schema-mapping` (XML/XSD-Option[255]) abhängig, welches Mapping verwendet wird. Falls kein Katalog-Mapping[48] vorhanden ist, wird das `schemaLocation` Attribut verwendet.
- `license-namespace-only`: Der Namespace wird importiert. Kein Schema-Dokument wird importiert.

▼ schema-location-hints

`--schema-location-hints =` `load-by-schemalocation | load-by-namespace | load-combining-both | ignore`
Definiert das Verhalten der Attribute `xsi:schemaLocation` und `xsi:noNamespaceSchemaLocation`: Legt fest, ob ein Schema-Dokument geladen werden soll und falls ja, anhand welcher Informationen es gesucht werden soll. Standardeinstellung: `load-by-schemalocation`.

- Der Wert `load-by-schemalocation` verwendet die URL des Schemapfads[422] in den Attributen `xsi:schemaLocation` und `xsi:noNamespaceSchemaLocation` in XML-Instanzdokumenten. Dies ist der **Standardwert**.
- Der Wert `load-by-namespace` verwendet den Namespace-Teil[422] von `xsi:schemaLocation` und - im Fall von `xsi:noNamespaceSchemaLocation` - einen leeren String und ermittelt das Schema über ein Katalog-Mapping[48].
- Bei Verwendung von `load-combining-both` und wenn entweder der Namespace-Teil oder der URL-Teil ein Katalog-Mapping[48] hat, so wird das Katalog-Mapping[48] verwendet. Wenn beide Katalog-Mappings[48] haben, ist es vom Wert der `--schema-mapping` Option (XML/XSD-Option[255]) abhängig, welches Mapping verwendet wird. Wenn weder der Namespace noch die URL ein Katalog-Mapping hat, wird die URL verwendet.
- Wenn der Wert der Option `ignore` ist, werden die beiden Attribute `xsi:schemaLocation` und `xsi:noNamespaceSchemaLocation` ignoriert.

▼ schema-mapping

`--schema-mapping =` `prefer-schemalocation | prefer-namespace`
Wenn ein Schema-Dokument sowohl anhand des Schema-Pfads als auch des Namespace gesucht werden soll, wird damit festgelegt, welche der beiden Optionen beim Katalog-Lookup Vorrang erhält. (Wenn eine der Optionen `--schemalocation-hints` oder `--schema-imports` einen Wert `load-`

`combining-both` hat und wenn die betroffenen Namespace- und URL-Teile beide Katalog-Mappings[48] haben, gibt der Wert dieser Option an, welches der beiden Mappings verwendet werden soll (das Namespace Mapping oder das URL-Mapping; der Wert `prefer-schemalocation` bezieht sich auf das URL-Mapping)). Der Standardwert ist `prefer-schemalocation`.

- xinclude

  `--xinclude =` `true|false`
  Aktiviert die Unterstützung für XML-Inkludierungen (XInclude). Der Standardwert ist `false`. Bei `false` werden die `include`-Elemente von XInclude ignoriert.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- xml-mode

  `--xml-mode =` `wf|id|valid`
  Definiert, welcher XML-Verarbeitungsmodus für das XML-Instanzdokument verwendet werden soll: `wf`=Wohlgeformtheitsprüfung; `id`=Wohlgeformtheitsprüfung mit ID/IDREF-Prüfung; `valid`=Validierung. Der Standardwert ist `wf`. Beachten Sie, dass jedes bei der Verarbeitung geladene Instanzdokument eine DTD referenzieren muss, damit der Wert `valid` ausgegeben werden kann. Falls keine DTD vorhanden ist, wird ein Fehler ausgegeben.

- xml-mode-for-schemas

  `--xml-mode-for-schemas =` `wf|id|valid`
  Definiert, welcher XML-Verarbeitungsmodus für XML-Schema-Dokumente verwendet werden soll: `wf`=Wohlgeformtheitsprüfung; `id`=Wohlgeformtheitsprüfung mit ID/IDREF-Prüfung; `valid`=Validierung. Der Standardwert ist `wf`. Beachten Sie, dass jedes bei der Verarbeitung geladene Schema-Dokument eine DTD referenzieren muss, damit der Wert `valid` ausgegeben werden kann. Falls keine DTD vorhanden ist, wird ein Fehler ausgegeben.

- xml-validation-error-as-warning

  `--xml-validation-error-as-warning =` `true|false`
  Bei `true` werden Validierungsfehler als Warnungen behandelt. Wenn Fehler als Warnungen behandelt werden, wird die weitere Verarbeitung, z.B. eine XSLT-Transformation ungeachtet der Fehler fortgesetzt. Die Standardeinstellung ist `false`.

- xsd

  `--xsd =` *`FILE`*
  Definiert ein oder mehrere XML-Schema-Dokumente, die für die Validierung von XML-Instanzdokumenten verwendet werden sollen. Um mehr als ein Schema-Dokument zu definieren, fügen Sie die Option mehrmals hinzu.

- xsd-version

  `--xsd-version =` `1.0|1.1|detect`
  Definiert die zu verwendende W3C Schema Definition Language (XSD) Version. Der Standardwert ist `1.0`. Diese Option eignet sich auch, um herauszufinden, inwiefern ein 1.0-kompatibles Schema nicht mit Schemaversion 1.1 kompatibel ist. Die Option `detect` ist eine Altova-spezifische Funktionalität. Mit dieser Option kann die Version des XML-Schema-Dokuments (`1.0` oder `1.1`) durch Lesen des Werts des `vc:minVersion` Attributs des `<xs:schema>` Elements des Dokuments ermittelt werden. Wenn der Wert des `@vc:minVersion` Attributs `1.1` ist, wird das Schema als Version `1.1` erkannt. Bei jedem anderen Wert wird das Schema als `1.0` erkannt. Bei jedem anderen Wert oder bei Fehlen

des `@vc:minVersion` Attributs wird das Schema als Version `1.0` gelesen.

- Kataloge und globale Ressourcen
  - catalog

    `--catalog =` *`FILE`*
    Gibt den absoluten Pfad zu einer Root-Katalog-Datei an, die nicht die installierte Root-Katalog-Datei ist. Der Standardwert ist der absolute Pfad zur installierten Root-Katalog-Datei. (`<installationsordner>\Altova\RaptorXMLServer2025\etc\RootCatalog.xml`). Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge[48].

  - user-catalog

    `--user-catalog =` *`FILE`*
    Definiert den absoluten Pfad zu einem XML-Katalog, der zusätzlich zum Root-Katalog verwendet werden soll. Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge[48].

  - enable-globalresources

    `--enable-globalresources =` `true|false`
    Aktiviert die globalen Ressourcen[55]. Standardwert ist `false`.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

  - globalresourceconfig [gc]

    `--gc | --globalresourceconfig =` *`VALUE`*
    Definiert die aktive Konfiguration der globalen Ressource[55] (und aktiviert globale Ressourcen)[55].

  - globalresourcefile [gr]

    `--gr | --globalresourcefile =` *`FILE`*
    Definiert die globale Ressourcendatei[55] (und aktiviert globale Ressourcen)[55].

- Erweiterungen

  Diese Optionen definieren die Behandlung von speziellen Erweiterungsfunktionen, die in einer Reihe von Enterprise Versionen von Altova-Produkten (wie z.B. in XMLSpy Enterprise Edition) verfügbar sind. Die Verwendung dieser Funktionen ist im Benutzerhandbuch des jeweiligen Produkts beschrieben.
  - chartext-disable

    `--chartext-disable =` `true|false`
    Deaktiviert Diagrammerweiterungen. Der Standardwert ist `false`.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

  - dotnetext-disable

    `--dotnetext-disable =` `true|false`
    Deaktiviert .NET-Erweiterungen. Der Standardwert ist `false`.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert

definiert wird.

- jvm-location

  --jvm-location = *FILE*
  FILE definiert den Pfad zur Java Virtual Machine (DLL unter Windows, freigegebenes Objekt unter Linux). Sie benötigen JVM, wenn Sie Java-Erweiterungsfunktionen[530] in Ihrem XSLT/XQuery-Code verwenden. Die Standardeinstellung ist `false`.

- javaext-barcode-location

  --javaext-barcode-location = *FILE*
  Definiert den Pfad zum Ordner, der die Barcode-Erweiterungsdatei `AltovaBarcodeExtension.jar` enthält. Der Pfad muss in einer der folgenden Formen angegeben werden:

  - als Datei-URI, z.B.: `--javaext-barcode-location="file:///C:/Program Files/Altova/RaptorXMLServer2025/etc/jar/"`
  - als Windows-Pfad mit maskierten umgekehrten Schrägstrichen, z.B.: `--javaext-barcode-location="C:\\Program Files\\Altova\\RaptorXMLServer2025\\etc\\jar\\"`

- javaext-disable

  --javaext-disable = `true|false`
  Deaktiviert Java-Erweiterungen. Der Standardwert ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- Allgemeine Optionen

  - error-format

    --error-format = `text|shortxml|longxml`
    Definiert das Format der Fehlerausgabe. Der Standardwert ist `text`. Mit den anderen Optionen werden XML-Formate generiert, wobei mit `longxml` mehr Details generiert werden.

  - error-limit

    --error-limit = *N | unlimited*
    Definiert das Fehlerlimit mit einem Wertebereich von `1` bis `9999` oder `unbegrenzt`. Der Standardwert ist `100`. Bei Erreichung des Fehlerlimits wird die Validierung gestoppt. Dient dazu, die Prozessorverwendung während der Validierung/Transformation einzuschränken.

  - info-limit

    --info-limit = *N | unlimited*
    Definiert die Grenze für Informationsmeldungen im Bereich von `1-65535` oder als `unlimited` (unbegrenzt). Die Verarbeitung wird auch nach Erreichen des Info-Limits fortgesetzt, doch werden weitere Meldungen nicht mehr ausgegeben. Der Standardwert ist `100`.

  - help

    --help
    Zeigt den Hilfetext zum Befehl an. Beispiel: `valany --h`. (Alternativ dazu kann der Befehl `help`

zusammen mit einem Argument verwendet werden. Beispiel: `help valany`.)

- listfile

  `--listfile =` `true|false`
  Bei `true` wird das Argument *`InputFile`* des Befehls als Textdatei behandelt, die einen Dateinamen pro Zeile enthält. Der Standardwert ist `false`. (Als Alternative können die Dateien im CLI getrennt durch ein Leerzeichen aufgelistet werden. Beachten Sie allerdings, dass CLIs eine maximale Zeichenanzahl haben). Beachten Sie, dass die Option `--listfile` nur auf Argumente, nicht aber auf Optionen angewendet wird.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- log-output

  `--log-output =` *`FILE`*
  Schreibt die Meldungsausgabe in die angegebene URL. Stellen Sie sicher, dass das CLI Schreibrechte für den Ausgabepfad hat.

- network-timeout

  `--network-timeout =` *`VALUE`*
  Definiert das Timeout für entfernte I/O-Operationen in Millisekunden. Der Standardwert ist: `40000`.

- recurse

  `--recurse =` `true|false`
  Dient zur Auswahl von Dateien innerhalb von Unterverzeichnissen einschließlich ZIP-Archiven. Bei `true` wählt das Argument *`InputFile`* des Befehls die angegebene Datei auch in den Unterverzeichnissen aus. Beispiel: `"test.zip|zip\test.xml"` wählt Dateien mit dem Namen `test.xml` auf allen Ordnerebenen des ZIP-Ordners aus. Referenzen auf ZIP-Dateien müssen in Anführungszeichen angegeben werden. Es können die Platzhalter `*` und `?` verwendet werden. Mit `*.xml` werden folglich alle `.xml` Dateien im (ZIP-)Ordner ausgewählt. Der Standardwert der Option ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- verbose

  `--verbose =` `true|false`
  Mit dem Wert `true` wird die Ausgabe zusätzlicher Informationen bei der Validierung aktiviert. Der Standardwert ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- verbose-output

  `--verbose-output =` *`FILE`*
  Schreibt die ausführliche Ausgabe in *`FILE`*.

- version

  `--version`

Zeigt die Version von RaptorXML Server an. Setzen Sie `--version` bei Verwendung mit einem Befehl vor den Befehl.

- warning-limit

  `--warning-limit =` *`N | unlimited`*
  Definiert das Warnungslimit im Bereich von `1-65535` oder `unbegrenzt`. Bei Erreichen dieses Limits wird die Verarbeitung fortgesetzt, doch werden keine weiteren Warnungen mehr ausgegeben. Der Standardwert ist `100`.

5.4.2 valxslt

Der Befehl **`valxslt`** erhält als einziges Argument eine XSLT-Datei und validiert diese.

```
raptorxml valxslt [options] XSLT-File
```

- Das Argument ***`XSLT-File`*** ist der Pfad und Name der zu validierenden XSLT-Datei.
- Die Validierung kann anhand der XSLT 1.0, 2.0 oder 3.0-Spezifikation erfolgen. Standardmäßig wird die XSLT 3.0-Spezifikation verwendet.

Beispiele

Beispiele für den Befehl **`valxslt`**:

- **`raptorxml`** `valxslt c:\Test.xslt`
- **`raptorxml`** `valxslt --xslt-version=2 c:\Test.xslt`

- Groß- und Kleinschreibung und Schrägstriche in der Befehlszeile

  `RaptorXML` (und **`RaptorXMLServer`** für Administrator-Befehle) *unter Windows*
  `raptorxml` und **`raptorxmlserver`** für Administrator-Befehle) *unter Windows und Unix (Linux, Mac)*

  * Beachten Sie, dass klein geschriebene Befehle (`raptorxml` und `raptorxmlserver`) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (`RaptorXML`) nur unter Windows und Mac ausgeführt werden.
  *Verwenden Sie auf Linux und Mac-Systemen Schrägstriche und auf Windows-Systemen umgekehrte Schrägstriche.

- Umgekehrte Schrägstriche, Leerzeichen und Sonderzeichen auf Windows-Systemen

  Auf Windows-Systemen: Verwenden Sie bei Vorhandensein von Leerzeichen oder Sonerzeichen in Strings (z.B. in Datei- oder Ordnernamen oder Firmen-, Personen- oder Produktnamen) Anführungszeichen: z.B., **`"Meine Datei"`**. Beachten Sie jedoch, dass ein von einem doppelten Anführungszeichen gefolgter umgekehrter Schrägstrich (z.B: **`"C:\Mein Verzeichnis\"`**) eventuell nicht korrekt gelesen wird, da der umgekehrte Schrägstrich auch den Beginn einer Escape-Sequenz markiert und die Escape-Sequenz **`\"`** für ein doppeltes Anführungszeichen steht. Wenn Sie diese Zeichensequenz mit einem Escape versehen wollen, verwenden Sie einen vorangestellten umgekehrten Schrägstrich, wie

den folgenden: `\\"`. Einfacher ausgedrückt: Wenn Sie einen Dateipfad, der Leerzeichen oder einen umgekehrten Schrägstrich am Ende enthält, schreiben müssen, so schreiben Sie diesen folgendermaßen: `"C:\Mein Verzeichnis\\"`.

Optionen

Optionen werden in ihrer kurzen Form (falls verfügbar) und in ihrer langen Form aufgelistet. Für die kurze und die lange Form können ein oder zwei Bindestriche verwendet werden. Eine Option kann, muss aber keinen Wert erhalten. Eine Option, die einen Wert erhält, wird folgendermaßen geschrieben: **`--option=wert`**. Werte können außer in zwei Fällen ohne Anführungszeichen definiert werden: (i) wenn der Wertestring Leerzeichen enthält oder (ii) wenn in der Beschreibung der Option explizit erwähnt ist, dass Anführungszeichen zwingend erforderlich sind. Wenn eine Option einen Booleschen Wert erhält und kein Wert definiert ist, so ist der Standardwert der Option `TRUE`. Mit Hilfe der Option **`--h, --help`** können Sie Informationen über den Befehl anzeigen.

- XSLT-Verarbeitung
  - initial-mode, template-mode

    `--initial-mode, --template-mode =` *VALUE*
    Definiert den Vorlagenmodus für die Transformation.

  - initial-template, template-entry-point

    `--initial-template, --template-entry-point =` *VALUE*
    Gibt den Namen einer benannten Vorlage im XSLT-Stylesheet an, das der Eintrittspunkt der Transformation ist.

  - xslt-version

    `--xslt-version =` `1|1.0|2|2.0|3|3.0|3.1`
    Definiert, ob der XSLT-Prozessor XSLT 1.0, XSLT 2.0 oder XSLT 3.0 verwenden soll. Der Standardwert ist `3`.

- XML-Schema und XML-Instanz
  - load-xml-with-psvi

    `--load-xml-with-psvi =` `true|false`
    Ermöglicht die Validierung von XML-Input-Dateien und die Generierung von Informationen für diese Dateien nach Validierung des Schemas. Der Standardwert ist: `true`.

  - schema-imports

    `--schema-imports =` `load-by-schemalocation | load-preferring-schemalocation | load-by-namespace | load-combining-both | license-namespace-only`
    Definiert das Verhalten von `xs:import` Elementen, von denen jedes ein optionales `namespace` Attribut und ein optionales `schemaLocation` Attribut hat: `<import namespace="someNS" schemaLocation="someURL">`. Mit der Option wird definiert, ob ein Schema-Dokument geladen oder nur ein Namespace lizenziert werden soll und, wenn ein Schema-Dokument geladen werden soll, anhand welcher Informationen es gesucht werden soll. Standardeinstellung: `load-preferring-schemalocation`. Das Verhalten ist das folgende:

- `load-by-schemalocation`: Anhand des Werts des `schemaLocation` Attributs wird der Schemapfad ermittelt, wobei Katalog-Mappings[48] berücksichtigt werden. Wenn das Namespace-Attribut vorhanden ist, wird der Namespace importiert (lizenziert).
- `load-preferring-schemalocation`: Wenn das `schemaLocation` Attribut vorhanden ist, wird es verwendet, wobei Katalog-Mappings[48] berücksichtigt werden. Falls kein `schemaLocation` Attribut vorhanden ist, wird der Wert des `namespace` Attributs über ein Katalog-Mapping[48] verwendet. Dies ist der **Standardwert**.
- `load-by-namespace`: Anhand des Werts des `namespace` Attributs wird der Schemapfad über ein Katalog-Mapping[48] ermittelt.
- `load-combining-both`: Wenn entweder das Attribut `namespace` oder das Attribut `schemaLocation` ein Katalog-Mapping[48] hat, so wird das Mapping verwendet. Wenn beide Attribute Katalog-Mappings[48] haben, ist es vom Wert der Option `--schema-mapping` (XML/XSD-Option[255]) abhängig, welches Mapping verwendet wird. Falls kein Katalog-Mapping[48] vorhanden ist, wird das `schemaLocation` Attribut verwendet.
- `license-namespace-only`: Der Namespace wird importiert. Kein Schema-Dokument wird importiert.

▼ schema-location-hints

`--schema-location-hints =` `load-by-schemalocation | load-by-namespace | load-combining-both | ignore`
Definiert das Verhalten der Attribute `xsi:schemaLocation` und `xsi:noNamespaceSchemaLocation`: Legt fest, ob ein Schema-Dokument geladen werden soll und falls ja, anhand welcher Informationen es gesucht werden soll. Standardeinstellung: `load-by-schemalocation`.

- Der Wert `load-by-schemalocation` verwendet die URL des Schemapfads[422] in den Attributen `xsi:schemaLocation` und `xsi:noNamespaceSchemaLocation` in XML-Instanzdokumenten. Dies ist der **Standardwert**.
- Der Wert `load-by-namespace` verwendet den Namespace-Teil[422] von `xsi:schemaLocation` und - im Fall von `xsi:noNamespaceSchemaLocation` - einen leeren String und ermittelt das Schema über ein Katalog-Mapping[48].
- Bei Verwendung von `load-combining-both` und wenn entweder der Namespace-Teil oder der URL-Teil ein Katalog-Mapping[48] hat, so wird das Katalog-Mapping[48] verwendet. Wenn beide Katalog-Mappings[48] haben, ist es vom Wert der `--schema-mapping` Option (XML/XSD-Option[255]) abhängig, welches Mapping verwendet wird. Wenn weder der Namespace noch die URL ein Katalog-Mapping hat, wird die URL verwendet.
- Wenn der Wert der Option `ignore` ist, werden die beiden Attribute `xsi:schemaLocation` und `xsi:noNamespaceSchemaLocation` ignoriert.

▼ schema-mapping

`--schema-mapping =` `prefer-schemalocation | prefer-namespace`
Wenn ein Schema-Dokument sowohl anhand des Schema-Pfads als auch des Namespace gesucht werden soll, wird damit festgelegt, welche der beiden Optionen beim Katalog-Lookup Vorrang erhält. (Wenn eine der Optionen `--schemalocation-hints` oder `--schema-imports` einen Wert `load-combining-both` hat und wenn die betroffenen Namespace- und URL-Teile beide Katalog-Mappings[48] haben, gibt der Wert dieser Option an, welches der beiden Mappings verwendet werden soll (das Namespace Mapping oder das URL-Mapping; der Wert `prefer-schemalocation` bezieht sich auf das URL-Mapping)). Der Standardwert ist `prefer-schemalocation`.

▼ xinclude

--xinclude = true|false
Aktiviert die Unterstützung für XML-Inkludierungen (XInclude). Der Standardwert ist false. Bei false werden die include-Elemente von XInclude ignoriert.
Hinweis: Die Booleschen Optionswerte werden auf true gesetzt, wenn die Option ohne einen Wert definiert wird.

- xml-mode

  --xml-mode = wf|id|valid
  Definiert, welcher XML-Verarbeitungsmodus für das XML-Instanzdokument verwendet werden soll: wf=Wohlgeformtheitsprüfung; id=Wohlgeformtheitsprüfung mit ID/IDREF-Prüfung; valid=Validierung. Der Standardwert ist wf. Beachten Sie, dass jedes bei der Verarbeitung geladene Instanzdokument eine DTD referenzieren muss, damit der Wert valid ausgegeben werden kann. Falls keine DTD vorhanden ist, wird ein Fehler ausgegeben.

- xml-mode-for-schemas

  --xml-mode-for-schemas = wf|id|valid
  Definiert, welcher XML-Verarbeitungsmodus für XML-Schema-Dokumente verwendet werden soll: wf=Wohlgeformtheitsprüfung; id=Wohlgeformtheitsprüfung mit ID/IDREF-Prüfung; valid=Validierung. Der Standardwert ist wf. Beachten Sie, dass jedes bei der Verarbeitung geladene Schema-Dokument eine DTD referenzieren muss, damit der Wert valid ausgegeben werden kann. Falls keine DTD vorhanden ist, wird ein Fehler ausgegeben.

- xpath-static-type-errors-as-warnings

  --xpath-static-type-errors-as-warnings = true|false
  Bei true werden alle im statischen XPath-Kontext gefundenen Typ-Fehler auf eine Warnung herabgestuft. Während die Ausführung bei einem Fehler fehlschlagen würde, könnte sie bei einer Warnung fortgesetzt werden. Die Standardeinstellung ist false.

- xsd-version

  --xsd-version = 1.0|1.1|detect
  Definiert die zu verwendende W3C Schema Definition Language (XSD) Version. Der Standardwert ist 1.0. Diese Option eignet sich auch, um herauszufinden, inwiefern ein 1.0-kompatibles Schema nicht mit Schemaversion 1.1 kompatibel ist. Die Option detect ist eine Altova-spezifische Funktionalität. Mit dieser Option kann die Version des XML-Schema-Dokuments (1.0 oder 1.1) durch Lesen des Werts des vc:minVersion Attributs des <xs:schema> Elements des Dokuments ermittelt werden. Wenn der Wert des @vc:minVersion Attributs 1.1 ist, wird das Schema als Version 1.1 erkannt. Bei jedem anderen Wert wird das Schema als 1.0 erkannt. Bei jedem anderen Wert oder bei Fehlen des @vc:minVersion Attributs wird das Schema als Version 1.0 gelesen.

- Kataloge und globale Ressourcen

  - catalog

    --catalog = *FILE*
    Gibt den absoluten Pfad zu einer Root-Katalog-Datei an, die nicht die installierte Root-Katalog-Datei ist. Der Standardwert ist der absolute Pfad zur installierten Root-Katalog-Datei. (<installationsordner>\Altova\RaptorXMLServer2025\etc\RootCatalog.xml). Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge[48].

- user-catalog

  --user-catalog = *FILE*
  Definiert den absoluten Pfad zu einem XML-Katalog, der zusätzlich zum Root-Katalog verwendet werden soll. Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge[48].

- enable-globalresources

  --enable-globalresources = `true|false`
  Aktiviert die globalen Ressourcen[55]. Standardwert ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- globalresourceconfig [gc]

  --gc | --globalresourceconfig = *VALUE*
  Definiert die aktive Konfiguration der globalen Ressource[55] (und aktiviert globale Ressourcen)[55].

- globalresourcefile [gr]

  --gr | --globalresourcefile = *FILE*
  Definiert die globale Ressourcendatei[55] (und aktiviert globale Ressourcen)[55].

- Erweiterungen

  Diese Optionen definieren die Behandlung von speziellen Erweiterungsfunktionen, die in einer Reihe von Enterprise Versionen von Altova-Produkten (wie z.B. in XMLSpy Enterprise Edition) verfügbar sind. Die Verwendung dieser Funktionen ist im Benutzerhandbuch des jeweiligen Produkts beschrieben.

  - chartext-disable

    --chartext-disable = `true|false`
    Deaktiviert Diagrammerweiterungen. Der Standardwert ist `false`.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

  - dotnetext-disable

    --dotnetext-disable = `true|false`
    Deaktiviert .NET-Erweiterungen. Der Standardwert ist `false`.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

  - jvm-location

    --jvm-location = *FILE*
    FILE definiert den Pfad zur Java Virtual Machine (DLL unter Windows, freigegebenes Objekt unter Linux). Sie benötigen JVM, wenn Sie Java-Erweiterungsfunktionen[530] in Ihrem XSLT/XQuery-Code verwenden. Die Standardeinstellung ist `false`.

  - javaext-barcode-location

    --javaext-barcode-location = *FILE*
    Definiert den Pfad zum Ordner, der die Barcode-Erweiterungsdatei `AltovaBarcodeExtension.jar` enthält. Der Pfad muss in einer der folgenden Formen angegeben werden:

- als Datei-URI, z.B.: `--javaext-barcode-location="file:///C:/Program Files/Altova/RaptorXMLServer2025/etc/jar/"`
- als Windows-Pfad mit maskierten umgekehrten Schrägstrichen, z.B.: `--javaext-barcode-location="C:\\Program Files\\Altova\\RaptorXMLServer2025\\etc\\jar\\"`

▼ javaext-disable

`--javaext-disable =` `true|false`
Deaktiviert Java-Erweiterungen. Der Standardwert ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

▼ Allgemeine Optionen

▼ error-format

`--error-format =` `text|shortxml|longxml`
Definiert das Format der Fehlerausgabe. Der Standardwert ist `text`. Mit den anderen Optionen werden XML-Formate generiert, wobei mit `longxml` mehr Details generiert werden.

▼ error-limit

`--error-limit =` *`N | unlimited`*
Definiert das Fehlerlimit mit einem Wertebereich von `1` bis `9999` oder `unbegrenzt`. Der Standardwert ist `100`. Bei Erreichung des Fehlerlimits wird die Validierung gestoppt. Dient dazu, die Prozessorverwendung während der Validierung/Transformation einzuschränken.

▼ info-limit

`--info-limit =` *`N | unlimited`*
Definiert die Grenze für Informationsmeldungen im Bereich von `1-65535` oder als `unlimited` (unbegrenzt). Die Verarbeitung wird auch nach Erreichen des Info-Limits fortgesetzt, doch werden weitere Meldungen nicht mehr ausgegeben. Der Standardwert ist `100`.

▼ help

`--help`
Zeigt den Hilfetext zum Befehl an. Beispiel: `valany --h`. (Alternativ dazu kann der Befehl `help` zusammen mit einem Argument verwendet werden. Beispiel: `help valany`.)

▼ listfile

`--listfile =` `true|false`
Bei `true` wird das Argument *`InputFile`* des Befehls als Textdatei behandelt, die einen Dateinamen pro Zeile enthält. Der Standardwert ist `false`. (Als Alternative können die Dateien im CLI getrennt durch ein Leerzeichen aufgelistet werden. Beachten Sie allerdings, dass CLIs eine maximale Zeichenanzahl haben). Beachten Sie, dass die Option `--listfile` nur auf Argumente, nicht aber auf Optionen angewendet wird.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- log-output

  --log-output = *FILE*
  Schreibt die Meldungsausgabe in die angegebene URL. Stellen Sie sicher, dass das CLI Schreibrechte für den Ausgabepfad hat.

- network-timeout

  --network-timeout = *VALUE*
  Definiert das Timeout für entfernte I/O-Operationen in Millisekunden. Der Standardwert ist: `40000`.

- recurse

  --recurse = `true|false`
  Dient zur Auswahl von Dateien innerhalb von Unterverzeichnissen einschließlich ZIP-Archiven. Bei `true` wählt das Argument *InputFile* des Befehls die angegebene Datei auch in den Unterverzeichnissen aus. Beispiel: `"test.zip|zip\test.xml"` wählt Dateien mit dem Namen `test.xml` auf allen Ordnerebenen des ZIP-Ordners aus. Referenzen auf ZIP-Dateien müssen in Anführungszeichen angegeben werden. Es können die Platzhalter `*` und `?` verwendet werden. Mit `*.xml` werden folglich alle `.xml` Dateien im (ZIP-)Ordner ausgewählt. Der Standardwert der Option ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- verbose

  --verbose = `true|false`
  Mit dem Wert `true` wird die Ausgabe zusätzlicher Informationen bei der Validierung aktiviert. Der Standardwert ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- verbose-output

  --verbose-output = *FILE*
  Schreibt die ausführliche Ausgabe in *FILE*.

- version

  --version
  Zeigt die Version von RaptorXML Server an. Setzen Sie `--version` bei Verwendung mit einem Befehl vor den Befehl.

- warning-limit

  --warning-limit = *N | unlimited*
  Definiert das Warnungslimit im Bereich von `1-65535` oder `unbegrenzt`. Bei Erreichen dieses Limits wird die Verarbeitung fortgesetzt, doch werden keine weiteren Warnungen mehr ausgegeben. Der Standardwert ist `100`.

5.5 JSON/Avro/YAML-Befehle

Mit Hilfe der JSON-Befehle können Sie die Gültigkeit und Wohlgeformtheit von JSON-Schema- und Instanzdokumetnen überprüfen. Diese Befehle sind unten aufgelistet und in den Unterabschnitten dieses Abschnitts näher beschrieben:

- avroextractschema 146: Extrahiert das Avro-Schema aus einer Avro-Binärdatei.
- json2xml 150: Konvertiert ein JSON-Instanzdokument in ein XML-Instanzdokument.
- jsonschema2xsd 155: Konvertiert ein JSON-Schema-Dokument in ein XML-Schema-Dokument.
- valavro 160: Validiert die Daten in einer oder mehreren Avro-Binärdateien anhand des Avro-Schemas der jeweiligen Binärdatei.
- valavrojson 164: Validiert eine oder mehrere JSON-Datendateien anhand eines Avro-Schemas,
- valavroschema 168: Validiert ein Avro-Schema anhand der Avro-Schema-Spezifikation.
- valjsonschema 172: Überprüft die Gültigkeit von JSON-Schema-Dokumenten.
- valjson 177: Überprüft die Gültigkeit von JSON-Dokumenten.
- valyaml 182: Überprüft die Gültigkeit von YAML-Dokumenten.
- wfjson 187: Überprüft die Wohlgeformtheit von JSON-Dokumenten.
- wfyaml 192: Überprüft die Wohlgeformtheit von YAML-Dokumenten.
- xml2json 195: Konvertiert ein XML-Instanzdokument in ein JSON-Instanzdokument.
- xsd2jsonschema 200: Konvertiert ein XML-Schema-Dokument in ein JSON-Schema-Dokument.

5.5.1 avroextractschema

Eine Avro-Binärdatei enthält einen Avro-Datenblock. Vor diesem Block steht das Avro-Schema, das die Struktur des Datenblocks definiert. Der Befehl **avroextractschema** extrahiert das Avro-Schema aus der Avro-Binärdatei und serialisiert das Avro-Schema als JSON.

```
raptorxml avroextractschema [options] --avrooutput=AvroSchemaFile AvroBinaryFile
```

- Das Argument *AvroBinaryFile* gibt die Avro-Binärdatei an, aus der das Avro-Schema extrahiert werden soll.
- Die Option `--avrooutput` definiert den Pfad zum extrahierten Avro-Schema.

Beispiel

Beispiel für den Befehl **avroextractschema**:

- **raptorxml** `avroextractschema --avrooutput=c:\MyAvroSchema.avsc c:\MyAvroBinary.avro`

▼ Groß- und Kleinschreibung und Schrägstriche in der Befehlszeile

RaptorXML (und **RaptorXMLServer** für Administrator-Befehle) *unter Windows*
raptorxml und **raptorxmlserver** für Administrator-Befehle) *unter Windows und Unix (Linux, Mac)*

* Beachten Sie, dass klein geschriebene Befehle (`raptorxml` und `raptorxmlserver`) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (`RaptorXML`) nur unter Windows und Mac ausgeführt werden.
*Verwenden Sie auf Linux und Mac-Systemen Schrägstriche und auf Windows-Systemen umgekehrte

Schrägstriche.

- Umgekehrte Schrägstriche, Leerzeichen und Sonderzeichen auf Windows-Systemen

  Auf Windows-Systemen: Verwenden Sie bei Vorhandensein von Leerzeichen oder Sonerzeichen in Strings (z.B. in Datei- oder Ordnernamen oder Firmen-, Personen- oder Produktnamen) Anführungszeichen: z.B., **`"Meine Datei"`**. Beachten Sie jedoch, dass ein von einem doppelten Anführungszeichen gefolgter umgekehrter Schrägstrich (z.B: **`"C:\Mein Verzeichnis\"`**) eventuell nicht korrekt gelesen wird, da der umgekehrte Schrägstrich auch den Beginn einer Escape-Sequenz markiert und die Escape-Sequenz **`\"`** für ein doppeltes Anführungszeichen steht. Wenn Sie diese Zeichensequenz mit einem Escape versehen wollen, verwenden Sie einen vorangestellten umgekehrten Schrägstrich, wie den folgenden: **`\\"`**. Einfacher ausgedrückt: Wenn Sie einen Dateipfad, der Leerzeichen oder einen umgekehrten Schrägstrich am Ende enthält, schreiben müssen, so schreiben Sie diesen folgendermaßen: **`"C:\Mein Verzeichnis\\"`**.

Optionen

Optionen werden in ihrer kurzen Form (falls verfügbar) und in ihrer langen Form aufgelistet. Für die kurze und die lange Form können ein oder zwei Bindestriche verwendet werden. Eine Option kann, muss aber keinen Wert erhalten. Eine Option, die einen Wert erhält, wird folgendermaßen geschrieben: **`--option=wert`**. Werte können außer in zwei Fällen ohne Anführungszeichen definiert werden: (i) wenn der Wertestring Leerzeichen enthält oder (ii) wenn in der Beschreibung der Option explizit erwähnt ist, dass Anführungszeichen zwingend erforderlich sind. Wenn eine Option einen Booleschen Wert erhält und kein Wert definiert ist, so ist der Standardwert der Option `TRUE`. Mit Hilfe der Option **`--h, --help`** können Sie Informationen über den Befehl anzeigen.

- Verarbeitung
  - output**`, avrooutput`**

    `--output =` *`FILE`*, **`--avrooutput =`** *`FILE`*
    Definiert den Pfad der Avro-Ausgabedatei.

  - recurse

    `--recurse =` `true|false`
    Dient zur Auswahl von Dateien innerhalb von Unterverzeichnissen einschließlich ZIP-Archiven. Bei `true` wählt das Argument *`InputFile`* des Befehls die angegebene Datei auch in den Unterverzeichnissen aus. Beispiel: `"test.zip|zip\test.xml"` wählt Dateien mit dem Namen `test.xml` auf allen Ordnerebenen des ZIP-Ordners aus. Referenzen auf ZIP-Dateien müssen in Anführungszeichen angegeben werden. Es können die Platzhalter `*` und `?` verwendet werden. Mit `*.xml` werden folglich alle `.xml` Dateien im (ZIP-)Ordner ausgewählt. Der Standardwert der Option ist `false`.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- Kataloge und globale Ressourcen
  - catalog

    `--catalog =` *`FILE`*
    Gibt den absoluten Pfad zu einer Root-Katalog-Datei an, die nicht die installierte Root-Katalog-Datei

ist. Der Standardwert ist der absolute Pfad zur installierten Root-Katalog-Datei. (`<installationsordner>\Altova\RaptorXMLServer2025\etc\RootCatalog.xml`). Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge[48].

- user-catalog

  `--user-catalog =` *`FILE`*
  Definiert den absoluten Pfad zu einem XML-Katalog, der zusätzlich zum Root-Katalog verwendet werden soll. Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge[48].

- enable-globalresources

  `--enable-globalresources =` `true|false`
  Aktiviert die globalen Ressourcen[55]. Standardwert ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- globalresourceconfig [gc]

  `--gc | --globalresourceconfig =` *`VALUE`*
  Definiert die aktive Konfiguration der globalen Ressource[55] (und aktiviert globale Ressourcen)[55].

- globalresourcefile [gr]

  `--gr | --globalresourcefile =` *`FILE`*
  Definiert die globale Ressourcendatei[55] (und aktiviert globale Ressourcen)[55].

Allgemeine Optionen

- error-format

  `--error-format =` `text|shortxml|longxml`
  Definiert das Format der Fehlerausgabe. Der Standardwert ist `text`. Mit den anderen Optionen werden XML-Formate generiert, wobei mit `longxml` mehr Details generiert werden.

- error-limit

  `--error-limit =` *`N | unlimited`*
  Definiert das Fehlerlimit mit einem Wertebereich von `1` bis `9999` oder `unbegrenzt`. Der Standardwert ist `100`. Bei Erreichung des Fehlerlimits wird die Validierung gestoppt. Dient dazu, die Prozessorverwendung während der Validierung/Transformation einzuschränken.

- info-limit

  `--info-limit =` *`N | unlimited`*
  Definiert die Grenze für Informationsmeldungen im Bereich von `1-65535` oder als `unlimited` (unbegrenzt). Die Verarbeitung wird auch nach Erreichen des Info-Limits fortgesetzt, doch werden weitere Meldungen nicht mehr ausgegeben. Der Standardwert ist `100`.

- help

  `--help`

Zeigt den Hilfetext zum Befehl an. Beispiel: `valany --h`. (Alternativ dazu kann der Befehl `help` zusammen mit einem Argument verwendet werden. Beispiel: `help valany`.)

- listfile

  `--listfile =` `true|false`
  Bei `true` wird das Argument *`InputFile`* des Befehls als Textdatei behandelt, die einen Dateinamen pro Zeile enthält. Der Standardwert ist `false`. (Als Alternative können die Dateien im CLI getrennt durch ein Leerzeichen aufgelistet werden. Beachten Sie allerdings, dass CLIs eine maximale Zeichenanzahl haben). Beachten Sie, dass die Option `--listfile` nur auf Argumente, nicht aber auf Optionen angewendet wird.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- log-output

  `--log-output =` *`FILE`*
  Schreibt die Meldungsausgabe in die angegebene URL. Stellen Sie sicher, dass das CLI Schreibrechte für den Ausgabepfad hat.

- network-timeout

  `--network-timeout =` *`VALUE`*
  Definiert das Timeout für entfernte I/O-Operationen in Millisekunden. Der Standardwert ist: `40000`.

- recurse

  `--recurse =` `true|false`
  Dient zur Auswahl von Dateien innerhalb von Unterverzeichnissen einschließlich ZIP-Archiven. Bei `true` wählt das Argument *`InputFile`* des Befehls die angegebene Datei auch in den Unterverzeichnissen aus. Beispiel: `"test.zip|zip\test.xml"` wählt Dateien mit dem Namen `test.xml` auf allen Ordnerebenen des ZIP-Ordners aus. Referenzen auf ZIP-Dateien müssen in Anführungszeichen angegeben werden. Es können die Platzhalter `*` und `?` verwendet werden. Mit `*.xml` werden folglich alle `.xml` Dateien im (ZIP-)Ordner ausgewählt. Der Standardwert der Option ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- verbose

  `--verbose =` `true|false`
  Mit dem Wert `true` wird die Ausgabe zusätzlicher Informationen bei der Validierung aktiviert. Der Standardwert ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- verbose-output

  `--verbose-output =` *`FILE`*
  Schreibt die ausführliche Ausgabe in *`FILE`*.

- version

--version
Zeigt die Version von RaptorXML Server an. Setzen Sie `--version` bei Verwendung mit einem Befehl vor den Befehl.

- warning-limit

  --warning-limit = *N | unlimited*
  Definiert das Warnungslimit im Bereich von `1-65535` oder `unbegrenzt`. Bei Erreichen dieses Limits wird die Verarbeitung fortgesetzt, doch werden keine weiteren Warnungen mehr ausgegeben. Der Standardwert ist `100`.

5.5.2 json2xml

Der Befehl **json2xml** konvertiert ein JSON-Instanzdokument in ein XML-Dokument.

```
raptorxml json2xml [options] JSONFile
```

- Das Argument *JSONFile* ist die zu konvertierende JSON-Datei.
- Mit Hilfe der Option `--conversion-output` definieren Sie den Pfad zur generierten XML-Datei.

Beispiel

Beispiel für den Befehl **json2xml**:

- **raptorxml** `json2xml --conversion-output=c:\MyXMLData.xml c:\MyJSONData.json`

- Groß- und Kleinschreibung und Schrägstriche in der Befehlszeile

  RaptorXML (und **RaptorXMLServer** für Administrator-Befehle) *unter Windows*
  raptorxml und **raptorxmlserver** für Administrator-Befehle) *unter Windows und Unix (Linux, Mac)*

  * Beachten Sie, dass klein geschriebene Befehle (`raptorxml` und `raptorxmlserver`) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (`RaptorXML`) nur unter Windows und Mac ausgeführt werden.
  *Verwenden Sie auf Linux und Mac-Systemen Schrägstriche und auf Windows-Systemen umgekehrte Schrägstriche.

- Umgekehrte Schrägstriche, Leerzeichen und Sonderzeichen auf Windows-Systemen

  Auf Windows-Systemen: Verwenden Sie bei Vorhandensein von Leerzeichen oder Sonerzeichen in Strings (z.B. in Datei- oder Ordnernamen oder Firmen-, Personen- oder Produktnamen) Anführungszeichen: z.B., **"Meine Datei"**. Beachten Sie jedoch, dass ein von einem doppelten Anführungszeichen gefolgter umgekehrter Schrägstrich (z.B: **"C:\Mein Verzeichnis\"**) eventuell nicht korrekt gelesen wird, da der umgekehrte Schrägstrich auch den Beginn einer Escape-Sequenz markiert und die Escape-Sequenz **\"** für ein doppeltes Anführungszeichen steht. Wenn Sie diese Zeichensequenz mit einem Escape versehen wollen, verwenden Sie einen vorangestellten umgekehrten Schrägstrich, wie den folgenden: **\\"**. Einfacher ausgedrückt: Wenn Sie einen Dateipfad, der Leerzeichen oder einen

umgekehrten Schrägstrich am Ende enthält, schreiben müssen, so schreiben Sie diesen folgendermaßen: `"C:\Mein Verzeichnis\\"`.

Optionen

Optionen werden in ihrer kurzen Form (falls verfügbar) und in ihrer langen Form aufgelistet. Für die kurze und die lange Form können ein oder zwei Bindestriche verwendet werden. Eine Option kann, muss aber keinen Wert erhalten. Eine Option, die einen Wert erhält, wird folgendermaßen geschrieben: **`--option=wert`**. Werte können außer in zwei Fällen ohne Anführungszeichen definiert werden: (i) wenn der Wertestring Leerzeichen enthält oder (ii) wenn in der Beschreibung der Option explizit erwähnt ist, dass Anführungszeichen zwingend erforderlich sind. Wenn eine Option einen Booleschen Wert erhält und kein Wert definiert ist, so ist der Standardwert der Option `TRUE`. Mit Hilfe der Option **`--h, --help`** können Sie Informationen über den Befehl anzeigen.

Optionen für die Konvertierung von JSON in XML

Mit diesen Optionen wird bei Konvertierungen zwischen XML und JSON die Behandlung bestimmter konvertierungsbezogener Details definiert.

array-element

`--array-element =` `VALUE`
Definiert den Namen des in ein Array-Element zu konvertierenden Elements.

attributes

`--attributes =` `true|false`
Bei **`true`** erfolgt eine Konvertierung zwischen XML-Attributen und JSON-Eigenschaften mit dem Präfix @. Andernfalls werden XML-Attribute und JSON-@-Eigenschaften nicht konvertiert. Der Standardwert ist **`true`**.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

comments

`--comments =` `true|false`
Bei **`true`** erfolgt eine Konvertierung zwischen XML-Kommentaren und JSON-Eigenschaften mit dem Präfix #. Andernfalls werden XML Attribute und JSON-#-Eigenschaften nicht konvertiert. Der Standardwert ist **`true`**.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

conversion-output, o

`--o, --conversion-output =` `FILE`
Definiert den Pfad und Namen der Datei, an die das Ergebnis der Konvertierung gesendet wird.

create-array-container

`--create-array-container =` `true|false`
Bei **`true`** wird für jedes JSON-Array im JSON-Quelldokument ein Container-Element in der generierten XML-Datei erstellt. Der Standardwert ist **`false`**.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert

definiert wird.

- encode-colons

  --encode-colons = `true|false`
  Bei **true** werden Doppelpunkte in JSON-Eigenschaftsnamen im generierten XML-Dokument kodiert. Der Standardwert ist **true**.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- json-type-hints

  --json-type-hints = `true|false`
  Bei **true** werden für Hinweise auf den Typ im JSON-Quelldokument im generierten XML-Dokument Attribute hinzugefügt. Der Standardwert ist **true**.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- pi

  --pi = `true|false`
  Bei **true** erfolgt eine Konvertierung zwischen XML Processing Instructions und JSON-Eigenschaften mit dem Präfix ?. Andernfalls werden XML-Attribute und JSON-?-Eigenschaften nicht konvertiert. Der Standardwert ist **true**.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- pretty-print

  --pp, --pretty-print = `true|false`
  Bei **true** wird das generierte Ausgabedokument mit Pretty-Print formatiert. Der Standardwert ist **false**.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- text

  --text = `true|false`
  Bei **true** erfolgt eine Konvertierung zwischen XML-Textinhalt und JSON-Eigenschaften mit dem Präfix $. Andernfalls werden XML-Attribute und JSON-$-Eigenschaften nicht konvertiert. Der Standardwert ist **true**.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- Allgemeine Optionen
  - error-format

    --error-format = `text|shortxml|longxml`
    Definiert das Format der Fehlerausgabe. Der Standardwert ist `text`. Mit den anderen Optionen werden XML-Formate generiert, wobei mit `longxml` mehr Details generiert werden.

- error-limit

  `--error-limit =` *`N | unlimited`*
  Definiert das Fehlerlimit mit einem Wertebereich von `1` bis `9999` oder `unbegrenzt`. Der Standardwert ist `100`. Bei Erreichung des Fehlerlimits wird die Validierung gestoppt. Dient dazu, die Prozessorverwendung während der Validierung/Transformation einzuschränken.

- info-limit

  `--info-limit =` *`N | unlimited`*
  Definiert die Grenze für Informationsmeldungen im Bereich von `1-65535` oder als `unlimited` (unbegrenzt). Die Verarbeitung wird auch nach Erreichen des Info-Limits fortgesetzt, doch werden weitere Meldungen nicht mehr ausgegeben. Der Standardwert ist `100`.

- help

  `--help`
  Zeigt den Hilfetext zum Befehl an. Beispiel: `valany --h`. (Alternativ dazu kann der Befehl `help` zusammen mit einem Argument verwendet werden. Beispiel: `help valany`.)

- listfile

  `--listfile =` `true|false`
  Bei `true` wird das Argument *`InputFile`* des Befehls als Textdatei behandelt, die einen Dateinamen pro Zeile enthält. Der Standardwert ist `false`. (Als Alternative können die Dateien im CLI getrennt durch ein Leerzeichen aufgelistet werden. Beachten Sie allerdings, dass CLIs eine maximale Zeichenanzahl haben). Beachten Sie, dass die Option `--listfile` nur auf Argumente, nicht aber auf Optionen angewendet wird.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- log-output

  `--log-output =` *`FILE`*
  Schreibt die Meldungsausgabe in die angegebene URL. Stellen Sie sicher, dass das CLI Schreibrechte für den Ausgabepfad hat.

- network-timeout

  `--network-timeout =` *`VALUE`*
  Definiert das Timeout für entfernte I/O-Operationen in Millisekunden. Der Standardwert ist: `40000`.

- recurse

  `--recurse =` `true|false`
  Dient zur Auswahl von Dateien innerhalb von Unterverzeichnissen einschließlich ZIP-Archiven. Bei `true` wählt das Argument *`InputFile`* des Befehls die angegebene Datei auch in den Unterverzeichnissen aus. Beispiel: `"test.zip|zip\test.xml"` wählt Dateien mit dem Namen `test.xml` auf allen Ordnerebenen des ZIP-Ordners aus. Referenzen auf ZIP-Dateien müssen in Anführungszeichen angegeben werden. Es können die Platzhalter `*` und `?` verwendet werden. Mit `*.xml` werden folglich alle `.xml` Dateien im (ZIP-)Ordner ausgewählt. Der Standardwert der Option ist `false`.

Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- verbose

  `--verbose =` `true|false`
  Mit dem Wert `true` wird die Ausgabe zusätzlicher Informationen bei der Validierung aktiviert. Der Standardwert ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- verbose-output

  `--verbose-output =` *`FILE`*
  Schreibt die ausführliche Ausgabe in *`FILE`*.

- version

  `--version`
  Zeigt die Version von RaptorXML Server an. Setzen Sie `--version` bei Verwendung mit einem Befehl vor den Befehl.

- warning-limit

  `--warning-limit =` *`N | unlimited`*
  Definiert das Warnungslimit im Bereich von `1-65535` oder `unbegrenzt`. Bei Erreichen dieses Limits wird die Verarbeitung fortgesetzt, doch werden keine weiteren Warnungen mehr ausgegeben. Der Standardwert ist `100`.

- Kataloge und globale Ressourcen
  - catalog

    `--catalog =` *`FILE`*
    Gibt den absoluten Pfad zu einer Root-Katalog-Datei an, die nicht die installierte Root-Katalog-Datei ist. Der Standardwert ist der absolute Pfad zur installierten Root-Katalog-Datei. (`<installationsordner>\Altova\RaptorXMLServer2025\etc\RootCatalog.xml`). Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge 48.

  - user-catalog

    `--user-catalog =` *`FILE`*
    Definiert den absoluten Pfad zu einem XML-Katalog, der zusätzlich zum Root-Katalog verwendet werden soll. Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge 48.

  - enable-globalresources

    `--enable-globalresources =` `true|false`
    Aktiviert die globalen Ressourcen 55. Standardwert ist `false`.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

  - globalresourceconfig [gc]

--gc | --globalresourceconfig = *VALUE*
Definiert die aktive Konfiguration der globalen Ressource[55] (und aktiviert globale Ressourcen)[55].

- globalresourcefile [gr]

  --gr | --globalresourcefile = *FILE*
  Definiert die globale Ressourcendatei[55] (und aktiviert globale Ressourcen)[55].

5.5.3 jsonschema2xsd

Der Befehl **jsonschema2xsd** konvertiert ein JSON-Schema-Dokument in ein XML-Schema-Dokument, das den Vorgaben der W3C XSD 1.0- und 1.1-Spezifikation entspricht.

```
raptorxml jsonschema2xsd [options] JSONSchemaFile
```

- Das Argument *JSONSchemaFile* ist die zu konvertierende JSON-Schema-Datei.
- Mit Hilfe der Option `--schema-conversion-output` definieren Sie den Pfad zur generierten XSD-Datei.

Beispiel

Beispiel für den Befehl **jsonschema2xsd**:

- **raptorxml** jsonschema2xsd --schema-conversion-output=c:\MyXMLSchema.xsd c:\MyJSONSchema.json

- Groß- und Kleinschreibung und Schrägstriche in der Befehlszeile

  RaptorXML (und **RaptorXMLServer** für Administrator-Befehle) *unter Windows*
  raptorxml und **raptorxmlserver** für Administrator-Befehle) *unter Windows und Unix (Linux, Mac)*

  * Beachten Sie, dass klein geschriebene Befehle (`raptorxml` und `raptorxmlserver`) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (`RaptorXML`) nur unter Windows und Mac ausgeführt werden.
  *Verwenden Sie auf Linux und Mac-Systemen Schrägstriche und auf Windows-Systemen umgekehrte Schrägstriche.

- Umgekehrte Schrägstriche, Leerzeichen und Sonderzeichen auf Windows-Systemen

  Auf Windows-Systemen: Verwenden Sie bei Vorhandensein von Leerzeichen oder Sonerzeichen in Strings (z.B. in Datei- oder Ordnernamen oder Firmen-, Personen- oder Produktnamen) Anführungszeichen: z.B., **"Meine Datei"**. Beachten Sie jedoch, dass ein von einem doppelten Anführungszeichen gefolgter umgekehrter Schrägstrich (z.B: **"C:\Mein Verzeichnis\"**) eventuell nicht korrekt gelesen wird, da der umgekehrte Schrägstrich auch den Beginn einer Escape-Sequenz markiert und die Escape-Sequenz **\"** für ein doppeltes Anführungszeichen steht. Wenn Sie diese Zeichensequenz mit einem Escape versehen wollen, verwenden Sie einen vorangestellten umgekehrten Schrägstrich, wie den folgenden: **\\"**. Einfacher ausgedrückt: Wenn Sie einen Dateipfad, der Leerzeichen oder einen umgekehrten Schrägstrich am Ende enthält, schreiben müssen, so schreiben Sie diesen

folgendermaßen: `"C:\Mein Verzeichnis\\"`.

Optionen

Optionen werden in ihrer kurzen Form (falls verfügbar) und in ihrer langen Form aufgelistet. Für die kurze und die lange Form können ein oder zwei Bindestriche verwendet werden. Eine Option kann, muss aber keinen Wert erhalten. Eine Option, die einen Wert erhält, wird folgendermaßen geschrieben: **`--option=wert`**. Werte können außer in zwei Fällen ohne Anführungszeichen definiert werden: (i) wenn der Wertestring Leerzeichen enthält oder (ii) wenn in der Beschreibung der Option explizit erwähnt ist, dass Anführungszeichen zwingend erforderlich sind. Wenn eine Option einen Booleschen Wert erhält und kein Wert definiert ist, so ist der Standardwert der Option `TRUE`. Mit Hilfe der Option **`--h, --help`** können Sie Informationen über den Befehl anzeigen.

JSON-Validierungsoptionen

Die folgenden Optionen stehen bei der Validierung des JSON-Quellschema-Dokuments zur Verfügung.

additional-schema

`--additional-schema =` *FILE*

Definiert URIs eines zusätzlichen Schema-Dokuments. Das zusätzliche Schema wird vom Hauptschema geladen und kann vom Hauptschema aus über die Eigenschaft **`id`** oder **`$id`** des zusätzlichen Schemas referenziert werden.

disable-format-checks

`--disable-format-checks =` `true|false`

Deaktiviert die durch das format-Attribut erzwungene semantische Validierung. Der Standardwert ist `false`.

Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

jsonschema-format

`--jsonschema-format =` `json|yaml`

Definiert das Format, in dem das JSON-Schema geschrieben ist: JSON oder YAML. Der Standardwert ist `json`.

jsonschema-version

`--jsonschema-version =` `draft04|draft06|draft07|2019-09|2020-12|oas-3.1|latest|detect`

Definiert, welche Version der JSON-Schema-Draft-Spezifikation verwendet werden soll. Der Standardwert ist `detect`.

strict-integer-checks

`--strict-integer-checks =` `true|false`

Gibt an, ob bei neueren Schemas, in denen die Ganzzahlüberprüfung weniger streng gehandhabt wird, die strengere Ganzzahlüberprüfung aus draft-04 verwendet werden soll. So ist z.B. `1.0` in draft-04 keine gültige Ganzzahl, in späteren Drafts jedoch schon. Diese Option hat keine Auswirkung auf draft-04-Schemas. Der Standardwert der Option ist `false`.

Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert

definiert wird.

- Konvertierung von JSON-Schema in XSD

  Mit diesen Optionen können Sie Details der Konvertierung von JSON-Schema in XSD definieren.

  - at-to-attributes

    `--at-to-attributes =` `true|false`
    Bei **`true`** werden Eigenschaften im JSON-Schema-Dokument mit dem Präfix @ im generierten XSD-Dokument in Attribute konvertiert. Der Standardwert ist **`true`**.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

  - consider-format

    `--consider-format =` `true|false`
    Wenn die Option auf **`true`** gesetzt ist, werden die Datentypen des Quellschemas, wenn möglich, in die entsprechenden Typen des Zielschemas konvertiert. Der Standardwert ist **`false`**.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

  - dollar-to-text

    `--dollar-to-text =` `true|false`
    Bei **`true`** werden Eigenschaften im JSON-Schema-Dokument mit dem Präfix $ im generierten XSD-Dokument in Text konvertiert. Der Standardwert ist **`true`**.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

  - ignore-comments

    `--ignore-comments =` `true|false`
    Bei **`true`** werden Eigenschaften namens `'#'` im JSON-Quellschemadokument ignoriert. Der Standardwert ist **`true`**.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

  - ignore-pi-proprties

    `--ignore-pi-properties =` `true|false`
    Bei **`true`** werden Eigenschaften im JSON-Quellschemadokument, die mit `'?'` beginnen, ignoriert. Der Standardwert ist **`true`**.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

  - ignore-xmlns-proprties

    `--ignore-xmlns-properties =` `true|false`
    Bei **`true`** werden Eigenschaften im JSON-Quellschemadokument, die mit `'@xmlns'` beginnen, ignoriert. Der Standardwert ist **`true`**.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- ignore-xsi-proprties

  --ignore-xsi-properties = `true|false`
  Bei **true** werden Eigenschaften im JSON-Quellschemadokument, die mit `'@xsi'` beginnen, ignoriert. Der Standardwert ist **true**.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- schema-conversion-output, o

  --o, --schema-converson-output = `FILE`
  Definiert den Pfad und Namen der Datei, an die das Ergebnis der Konvertierung gesendet wird.

- Allgemeine Optionen
  - error-format

    --error-format = `text|shortxml|longxml`
    Definiert das Format der Fehlerausgabe. Der Standardwert ist `text`. Mit den anderen Optionen werden XML-Formate generiert, wobei mit `longxml` mehr Details generiert werden.

  - error-limit

    --error-limit = *N | unlimited*
    Definiert das Fehlerlimit mit einem Wertebereich von `1` bis `9999` oder `unbegrenzt`. Der Standardwert ist `100`. Bei Erreichung des Fehlerlimits wird die Validierung gestoppt. Dient dazu, die Prozessorverwendung während der Validierung/Transformation einzuschränken.

  - info-limit

    --info-limit = *N | unlimited*
    Definiert die Grenze für Informationsmeldungen im Bereich von `1-65535` oder als `unlimited` (unbegrenzt). Die Verarbeitung wird auch nach Erreichen des Info-Limits fortgesetzt, doch werden weitere Meldungen nicht mehr ausgegeben. Der Standardwert ist `100`.

  - help

    --help
    Zeigt den Hilfetext zum Befehl an. Beispiel: `valany --h`. (Alternativ dazu kann der Befehl `help` zusammen mit einem Argument verwendet werden. Beispiel: `help valany`.)

  - listfile

    --listfile = `true|false`
    Bei `true` wird das Argument *InputFile* des Befehls als Textdatei behandelt, die einen Dateinamen pro Zeile enthält. Der Standardwert ist `false`. (Als Alternative können die Dateien im CLI getrennt durch ein Leerzeichen aufgelistet werden. Beachten Sie allerdings, dass CLIs eine maximale Zeichenanzahl haben). Beachten Sie, dass die Option `--listfile` nur auf Argumente, nicht aber auf Optionen angewendet wird.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert

definiert wird.

- log-output

  --log-output = *FILE*
  Schreibt die Meldungsausgabe in die angegebene URL. Stellen Sie sicher, dass das CLI Schreibrechte für den Ausgabepfad hat.

- network-timeout

  --network-timeout = *VALUE*
  Definiert das Timeout für entfernte I/O-Operationen in Millisekunden. Der Standardwert ist: `40000`.

- recurse

  --recurse = `true|false`
  Dient zur Auswahl von Dateien innerhalb von Unterverzeichnissen einschließlich ZIP-Archiven. Bei `true` wählt das Argument *InputFile* des Befehls die angegebene Datei auch in den Unterverzeichnissen aus. Beispiel: `"test.zip|zip\test.xml"` wählt Dateien mit dem Namen `test.xml` auf allen Ordnerebenen des ZIP-Ordners aus. Referenzen auf ZIP-Dateien müssen in Anführungszeichen angegeben werden. Es können die Platzhalter `*` und `?` verwendet werden. Mit `*.xml` werden folglich alle `.xml` Dateien im (ZIP-)Ordner ausgewählt. Der Standardwert der Option ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- verbose

  --verbose = `true|false`
  Mit dem Wert `true` wird die Ausgabe zusätzlicher Informationen bei der Validierung aktiviert. Der Standardwert ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- verbose-output

  --verbose-output = *FILE*
  Schreibt die ausführliche Ausgabe in *FILE*.

- version

  --version
  Zeigt die Version von RaptorXML Server an. Setzen Sie `--version` bei Verwendung mit einem Befehl vor den Befehl.

- warning-limit

  --warning-limit = *N | unlimited*
  Definiert das Warnungslimit im Bereich von `1-65535` oder `unbegrenzt`. Bei Erreichen dieses Limits wird die Verarbeitung fortgesetzt, doch werden keine weiteren Warnungen mehr ausgegeben. Der Standardwert ist `100`.

- Kataloge und globale Ressourcen
  - catalog

    --catalog = *FILE*
    Gibt den absoluten Pfad zu einer Root-Katalog-Datei an, die nicht die installierte Root-Katalog-Datei ist. Der Standardwert ist der absolute Pfad zur installierten Root-Katalog-Datei. (`<installationsordner>\Altova\RaptorXMLServer2025\etc\RootCatalog.xml`). Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge[48].

  - user-catalog

    --user-catalog = *FILE*
    Definiert den absoluten Pfad zu einem XML-Katalog, der zusätzlich zum Root-Katalog verwendet werden soll. Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge[48].

  - enable-globalresources

    --enable-globalresources = `true|false`
    Aktiviert die globalen Ressourcen[55]. Standardwert ist `false`.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

  - globalresourceconfig [gc]

    --gc | --globalresourceconfig = *VALUE*
    Definiert die aktive Konfiguration der globalen Ressource[55] (und aktiviert globale Ressourcen)[55].

  - globalresourcefile [gr]

    --gr | --globalresourcefile = *FILE*
    Definiert die globale Ressourcendatei[55] (und aktiviert globale Ressourcen)[55].

5.5.4 valavro (avro)

Der Befehl **`valavro | avro`** validiert den Datenblock in einer oder mehreren Avro-Binärdateien anhand der Avro-Schemas in der jeweiligen Binärdatei.

```
raptorxml valavro | avro [options] AvroBinaryFile
```

- Das Argument *`AvroBinaryFile`* definiert eine oder mehrere zu validierende Avro-Binärdateien. Dabei wird der Datenblock in den einzelnen Avro-Binärdateien anhand des Avro-Schemas in dieser Binärdatei validiert.
- Um mehrere Avro-Binärdateien zu validieren, (i) listen Sie entweder die zu validierenden Dateien im CLI auf, wobei jede Datei durch ein Leerzeichen von der nächsten getrennt wird, oder (ii) listen Sie die zu validierenden Dateien in einer Textdatei (`.txt`-Datei) auf (ein Dateiname pro Zeile) und geben Sie diese Textdatei zusammen mit der auf `true` gesetzten Option `--listfile`[252] (*siehe Liste der Optionen unten*) als das *`AvroBinaryFile`*-Argument an.

Beispiele

Beispiele für den Befehl **valavro**:

- **raptorxml** valavro c:\MyAvroBinary.avro
- **raptorxml** valavro c:\MyAvroBinary01.avro c:\MyAvroBinary02.avro
- **raptorxml** avro --listfile=true c:\MyFileList.txt

▼ Groß- und Kleinschreibung und Schrägstriche in der Befehlszeile

RaptorXML (und **RaptorXMLServer** für Administrator-Befehle) *unter Windows*
raptorxml und **raptorxmlserver** für Administrator-Befehle) *unter Windows und Unix (Linux, Mac)*

* Beachten Sie, dass klein geschriebene Befehle (raptorxml und raptorxmlserver) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (RaptorXML) nur unter Windows und Mac ausgeführt werden.
*Verwenden Sie auf Linux und Mac-Systemen Schrägstriche und auf Windows-Systemen umgekehrte Schrägstriche.

▼ Umgekehrte Schrägstriche, Leerzeichen und Sonderzeichen auf Windows-Systemen

Auf Windows-Systemen: Verwenden Sie bei Vorhandensein von Leerzeichen oder Sonerzeichen in Strings (z.B. in Datei- oder Ordnernamen oder Firmen-, Personen- oder Produktnamen) Anführungszeichen: z.B., **"Meine Datei"**. Beachten Sie jedoch, dass ein von einem doppelten Anführungszeichen gefolgter umgekehrter Schrägstrich (z.B: **"C:\Mein Verzeichnis\"**) eventuell nicht korrekt gelesen wird, da der umgekehrte Schrägstrich auch den Beginn einer Escape-Sequenz markiert und die Escape-Sequenz **\"** für ein doppeltes Anführungszeichen steht. Wenn Sie diese Zeichensequenz mit einem Escape versehen wollen, verwenden Sie einen vorangestellten umgekehrten Schrägstrich, wie den folgenden: **\\"**. Einfacher ausgedrückt: Wenn Sie einen Dateipfad, der Leerzeichen oder einen umgekehrten Schrägstrich am Ende enthält, schreiben müssen, so schreiben Sie diesen folgendermaßen: **"C:\Mein Verzeichnis\\"**.

Optionen

Optionen werden in ihrer kurzen Form (falls verfügbar) und in ihrer langen Form aufgelistet. Für die kurze und die lange Form können ein oder zwei Bindestriche verwendet werden. Eine Option kann, muss aber keinen Wert erhalten. Eine Option, die einen Wert erhält, wird folgendermaßen geschrieben: **--option=wert**. Werte können außer in zwei Fällen ohne Anführungszeichen definiert werden: (i) wenn der Wertestring Leerzeichen enthält oder (ii) wenn in der Beschreibung der Option explizit erwähnt ist, dass Anführungszeichen zwingend erforderlich sind. Wenn eine Option einen Booleschen Wert erhält und kein Wert definiert ist, so ist der Standardwert der Option TRUE. Mit Hilfe der Option **--h, --help** können Sie Informationen über den Befehl anzeigen.

▼ Verarbeitung

▼ listfile

--listfile = true|false
Bei true wird das Argument *InputFile* des Befehls als Textdatei behandelt, die einen Dateinamen pro Zeile enthält. Der Standardwert ist false. (Als Alternative können die Dateien im CLI getrennt durch ein Leerzeichen aufgelistet werden. Beachten Sie allerdings, dass CLIs eine maximale

Zeichenanzahl haben). Beachten Sie, dass die Option `--listfile` nur auf Argumente, nicht aber auf Optionen angewendet wird.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- recurse

  `--recurse =` `true|false`
  Dient zur Auswahl von Dateien innerhalb von Unterverzeichnissen einschließlich ZIP-Archiven. Bei `true` wählt das Argument *`InputFile`* des Befehls die angegebene Datei auch in den Unterverzeichnissen aus. Beispiel: `"test.zip|zip\test.xml"` wählt Dateien mit dem Namen `test.xml` auf allen Ordnerebenen des ZIP-Ordners aus. Referenzen auf ZIP-Dateien müssen in Anführungszeichen angegeben werden. Es können die Platzhalter `*` und `?` verwendet werden. Mit `*.xml` werden folglich alle `.xml` Dateien im (ZIP-)Ordner ausgewählt. Der Standardwert der Option ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- Kataloge und globale Ressourcen
  - catalog

    `--catalog =` *`FILE`*
    Gibt den absoluten Pfad zu einer Root-Katalog-Datei an, die nicht die installierte Root-Katalog-Datei ist. Der Standardwert ist der absolute Pfad zur installierten Root-Katalog-Datei. (`<installationsordner>\Altova\RaptorXMLServer2025\etc\RootCatalog.xml`). Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge[48].

  - user-catalog

    `--user-catalog =` *`FILE`*
    Definiert den absoluten Pfad zu einem XML-Katalog, der zusätzlich zum Root-Katalog verwendet werden soll. Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge[48].

  - enable-globalresources

    `--enable-globalresources =` `true|false`
    Aktiviert die globalen Ressourcen[55]. Standardwert ist `false`.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

  - globalresourceconfig [gc]

    `--gc | --globalresourceconfig =` *`VALUE`*
    Definiert die aktive Konfiguration der globalen Ressource[55] (und aktiviert globale Ressourcen)[55].

  - globalresourcefile [gr]

    `--gr | --globalresourcefile =` *`FILE`*
    Definiert die globale Ressourcendatei[55] (und aktiviert globale Ressourcen)[55].

- Allgemeine Optionen
  - error-format

    --error-format = text|shortxml|longxml
    Definiert das Format der Fehlerausgabe. Der Standardwert ist text. Mit den anderen Optionen werden XML-Formate generiert, wobei mit longxml mehr Details generiert werden.

  - error-limit

    --error-limit = *N | unlimited*
    Definiert das Fehlerlimit mit einem Wertebereich von 1 bis 9999 oder unbegrenzt. Der Standardwert ist 100. Bei Erreichung des Fehlerlimits wird die Validierung gestoppt. Dient dazu, die Prozessorverwendung während der Validierung/Transformation einzuschränken.

  - info-limit

    --info-limit = *N | unlimited*
    Definiert die Grenze für Informationsmeldungen im Bereich von 1-65535 oder als unlimited (unbegrenzt). Die Verarbeitung wird auch nach Erreichen des Info-Limits fortgesetzt, doch werden weitere Meldungen nicht mehr ausgegeben. Der Standardwert ist 100.

  - help

    --help
    Zeigt den Hilfetext zum Befehl an. Beispiel: valany --h. (Alternativ dazu kann der Befehl help zusammen mit einem Argument verwendet werden. Beispiel: help valany.)

  - listfile

    --listfile = true|false
    Bei true wird das Argument *InputFile* des Befehls als Textdatei behandelt, die einen Dateinamen pro Zeile enthält. Der Standardwert ist false. (Als Alternative können die Dateien im CLI getrennt durch ein Leerzeichen aufgelistet werden. Beachten Sie allerdings, dass CLIs eine maximale Zeichenanzahl haben). Beachten Sie, dass die Option --listfile nur auf Argumente, nicht aber auf Optionen angewendet wird.
    Hinweis: Die Booleschen Optionswerte werden auf true gesetzt, wenn die Option ohne einen Wert definiert wird.

  - log-output

    --log-output = *FILE*
    Schreibt die Meldungsausgabe in die angegebene URL. Stellen Sie sicher, dass das CLI Schreibrechte für den Ausgabepfad hat.

  - network-timeout

    --network-timeout = *VALUE*
    Definiert das Timeout für entfernte I/O-Operationen in Millisekunden. Der Standardwert ist: 40000.

  - recurse

    --recurse = true|false
    Dient zur Auswahl von Dateien innerhalb von Unterverzeichnissen einschließlich ZIP-Archiven. Bei

`true` wählt das Argument *`InputFile`* des Befehls die angegebene Datei auch in den Unterverzeichnissen aus. Beispiel: `"test.zip|zip\test.xml"` wählt Dateien mit dem Namen `test.xml` auf allen Ordnerebenen des ZIP-Ordners aus. Referenzen auf ZIP-Dateien müssen in Anführungszeichen angegeben werden. Es können die Platzhalter `*` und `?` verwendet werden. Mit `*.xml` werden folglich alle `.xml` Dateien im (ZIP-)Ordner ausgewählt. Der Standardwert der Option ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- verbose

  `--verbose =` `true|false`
  Mit dem Wert `true` wird die Ausgabe zusätzlicher Informationen bei der Validierung aktiviert. Der Standardwert ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- verbose-output

  `--verbose-output =` *`FILE`*
  Schreibt die ausführliche Ausgabe in *`FILE`*.

- version

  `--version`
  Zeigt die Version von RaptorXML Server an. Setzen Sie `--version` bei Verwendung mit einem Befehl vor den Befehl.

- warning-limit

  `--warning-limit =` *`N | unlimited`*
  Definiert das Warnungslimit im Bereich von `1-65535` oder `unbegrenzt`. Bei Erreichen dieses Limits wird die Verarbeitung fortgesetzt, doch werden keine weiteren Warnungen mehr ausgegeben. Der Standardwert ist `100`.

5.5.5 valavrojson (avrojson)

Der Befehl **`valavrojson | avro`** validiert ein JSON-Dokument anhand eines Avro-Schemas.

```
raptorxml valavrojson | avrojson [options] --avroschema=AvroSchema JSONFile
```

- Das Argument *`JSONFile`* definiert das zu validierende JSON-Dokument.
- Mit der Option **`--avroschema`** wird das Avro-Schema angegeben, anhand dessen das JSON-Dokument validiert werden soll.
- Um mehrere JSON-Dateien zu validieren, (i) listen Sie entweder die zu validierenden Dateien im CLI auf, wobei jede Datei durch ein Leerzeichen von der nächsten getrennt wird, oder (ii) listen Sie die zu validierenden Dateien in einer Textdatei (`.txt`-Datei) auf (ein Dateiname pro Zeile) und geben Sie diese

Textdatei zusammen mit der auf `true` gesetzten Option `--listfile` 252 (*siehe Liste der Optionen unten*) als das *`JSONFile`*-Argument an.

Beispiele

Beispiele für den Befehl **`valavrojson`**:

- **`raptorxml`** `valavrojson --avroschema=c:\MyAvroSchema.avsc c:\MyJSONDataFile.json`
- **`raptorxml`** `avrojson --avroschema=c:\MyAvroSchema.avsc c:\MyJSONDataFile.json`

▼ Groß- und Kleinschreibung und Schrägstriche in der Befehlszeile

`RaptorXML` (und **`RaptorXMLServer`** für Administrator-Befehle) *unter Windows*
`raptorxml` und **`raptorxmlserver`** für Administrator-Befehle) *unter Windows und Unix (Linux, Mac)*

* Beachten Sie, dass klein geschriebene Befehle (`raptorxml` und `raptorxmlserver`) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (`RaptorXML`) nur unter Windows und Mac ausgeführt werden.
*Verwenden Sie auf Linux und Mac-Systemen Schrägstriche und auf Windows-Systemen umgekehrte Schrägstriche.

▼ Umgekehrte Schrägstriche, Leerzeichen und Sonderzeichen auf Windows-Systemen

Auf Windows-Systemen: Verwenden Sie bei Vorhandensein von Leerzeichen oder Sonerzeichen in Strings (z.B. in Datei- oder Ordnernamen oder Firmen-, Personen- oder Produktnamen) Anführungszeichen: z.B., **`"Meine Datei"`**. Beachten Sie jedoch, dass ein von einem doppelten Anführungszeichen gefolgter umgekehrter Schrägstrich (z.B: **`"C:\Mein Verzeichnis\"`**) eventuell nicht korrekt gelesen wird, da der umgekehrte Schrägstrich auch den Beginn einer Escape-Sequenz markiert und die Escape-Sequenz **`\"`** für ein doppeltes Anführungszeichen steht. Wenn Sie diese Zeichensequenz mit einem Escape versehen wollen, verwenden Sie einen vorangestellten umgekehrten Schrägstrich, wie den folgenden: **`\\"`**. Einfacher ausgedrückt: Wenn Sie einen Dateipfad, der Leerzeichen oder einen umgekehrten Schrägstrich am Ende enthält, schreiben müssen, so schreiben Sie diesen folgendermaßen: **`"C:\Mein Verzeichnis\\"`**.

Optionen

Optionen werden in ihrer kurzen Form (falls verfügbar) und in ihrer langen Form aufgelistet. Für die kurze und die lange Form können ein oder zwei Bindestriche verwendet werden. Eine Option kann, muss aber keinen Wert erhalten. Eine Option, die einen Wert erhält, wird folgendermaßen geschrieben: **`--option=wert`**. Werte können außer in zwei Fällen ohne Anführungszeichen definiert werden: (i) wenn der Wertestring Leerzeichen enthält oder (ii) wenn in der Beschreibung der Option explizit erwähnt ist, dass Anführungszeichen zwingend erforderlich sind. Wenn eine Option einen Booleschen Wert erhält und kein Wert definiert ist, so ist der Standardwert der Option `TRUE`. Mit Hilfe der Option **`--h, --help`** können Sie Informationen über den Befehl anzeigen.

▼ Verarbeitung

▼ listfile

`--listfile =` `true|false`
Bei `true` wird das Argument *`InputFile`* des Befehls als Textdatei behandelt, die einen Dateinamen pro Zeile enthält. Der Standardwert ist `false`. (Als Alternative können die Dateien im CLI getrennt

durch ein Leerzeichen aufgelistet werden. Beachten Sie allerdings, dass CLIs eine maximale Zeichenanzahl haben). Beachten Sie, dass die Option `--listfile` nur auf Argumente, nicht aber auf Optionen angewendet wird.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- recurse

  `--recurse =` `true|false`
  Dient zur Auswahl von Dateien innerhalb von Unterverzeichnissen einschließlich ZIP-Archiven. Bei `true` wählt das Argument *`InputFile`* des Befehls die angegebene Datei auch in den Unterverzeichnissen aus. Beispiel: `"test.zip|zip\test.xml"` wählt Dateien mit dem Namen `test.xml` auf allen Ordnerebenen des ZIP-Ordners aus. Referenzen auf ZIP-Dateien müssen in Anführungszeichen angegeben werden. Es können die Platzhalter `*` und `?` verwendet werden. Mit `*.xml` werden folglich alle `.xml` Dateien im (ZIP-)Ordner ausgewählt. Der Standardwert der Option ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- Kataloge und globale Ressourcen
  - catalog

    `--catalog =` *`FILE`*
    Gibt den absoluten Pfad zu einer Root-Katalog-Datei an, die nicht die installierte Root-Katalog-Datei ist. Der Standardwert ist der absolute Pfad zur installierten Root-Katalog-Datei. (`<installationsordner>\Altova\RaptorXMLServer2025\etc\RootCatalog.xml`). Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge[48].

  - user-catalog

    `--user-catalog =` *`FILE`*
    Definiert den absoluten Pfad zu einem XML-Katalog, der zusätzlich zum Root-Katalog verwendet werden soll. Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge[48].

  - enable-globalresources

    `--enable-globalresources =` `true|false`
    Aktiviert die globalen Ressourcen[55]. Standardwert ist `false`.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

  - globalresourceconfig [gc]

    `--gc | --globalresourceconfig =` *`VALUE`*
    Definiert die aktive Konfiguration der globalen Ressource[55] (und aktiviert globale Ressourcen)[55].

  - globalresourcefile [gr]

    `--gr | --globalresourcefile =` *`FILE`*
    Definiert die globale Ressourcendatei[55] (und aktiviert globale Ressourcen)[55].

- Allgemeine Optionen
  - error-format

    `--error-format =` `text|shortxml|longxml`
    Definiert das Format der Fehlerausgabe. Der Standardwert ist `text`. Mit den anderen Optionen werden XML-Formate generiert, wobei mit `longxml` mehr Details generiert werden.

  - error-limit

    `--error-limit =` *`N | unlimited`*
    Definiert das Fehlerlimit mit einem Wertebereich von `1` bis `9999` oder `unbegrenzt`. Der Standardwert ist `100`. Bei Erreichung des Fehlerlimits wird die Validierung gestoppt. Dient dazu, die Prozessorverwendung während der Validierung/Transformation einzuschränken.

  - info-limit

    `--info-limit =` *`N | unlimited`*
    Definiert die Grenze für Informationsmeldungen im Bereich von `1-65535` oder als `unlimited` (unbegrenzt). Die Verarbeitung wird auch nach Erreichen des Info-Limits fortgesetzt, doch werden weitere Meldungen nicht mehr ausgegeben. Der Standardwert ist `100`.

  - help

    `--help`
    Zeigt den Hilfetext zum Befehl an. Beispiel: `valany --h`. (Alternativ dazu kann der Befehl `help` zusammen mit einem Argument verwendet werden. Beispiel: `help valany`.)

  - listfile

    `--listfile =` `true|false`
    Bei `true` wird das Argument *`InputFile`* des Befehls als Textdatei behandelt, die einen Dateinamen pro Zeile enthält. Der Standardwert ist `false`. (Als Alternative können die Dateien im CLI getrennt durch ein Leerzeichen aufgelistet werden. Beachten Sie allerdings, dass CLIs eine maximale Zeichenanzahl haben). Beachten Sie, dass die Option `--listfile` nur auf Argumente, nicht aber auf Optionen angewendet wird.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

  - log-output

    `--log-output =` *`FILE`*
    Schreibt die Meldungsausgabe in die angegebene URL. Stellen Sie sicher, dass das CLI Schreibrechte für den Ausgabepfad hat.

  - network-timeout

    `--network-timeout =` *`VALUE`*
    Definiert das Timeout für entfernte I/O-Operationen in Millisekunden. Der Standardwert ist: `40000`.

  - recurse

    `--recurse =` `true|false`

Dient zur Auswahl von Dateien innerhalb von Unterverzeichnissen einschließlich ZIP-Archiven. Bei `true` wählt das Argument *InputFile* des Befehls die angegebene Datei auch in den Unterverzeichnissen aus. Beispiel: `"test.zip|zip\test.xml"` wählt Dateien mit dem Namen `test.xml` auf allen Ordnerebenen des ZIP-Ordners aus. Referenzen auf ZIP-Dateien müssen in Anführungszeichen angegeben werden. Es können die Platzhalter `*` und `?` verwendet werden. Mit `*.xml` werden folglich alle `.xml` Dateien im (ZIP-)Ordner ausgewählt. Der Standardwert der Option ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- verbose

  `--verbose =` `true|false`
  Mit dem Wert `true` wird die Ausgabe zusätzlicher Informationen bei der Validierung aktiviert. Der Standardwert ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- verbose-output

  `--verbose-output =` *FILE*
  Schreibt die ausführliche Ausgabe in *FILE*.

- version

  `--version`
  Zeigt die Version von RaptorXML Server an. Setzen Sie `--version` bei Verwendung mit einem Befehl vor den Befehl.

- warning-limit

  `--warning-limit =` *N | unlimited*
  Definiert das Warnungslimit im Bereich von `1-65535` oder `unbegrenzt`. Bei Erreichen dieses Limits wird die Verarbeitung fortgesetzt, doch werden keine weiteren Warnungen mehr ausgegeben. Der Standardwert ist `100`.

5.5.6 valavroschema (avroschema)

Der Befehl **`valavroschema | avroschema`** validiert ein oder mehrere Avro-Schema-Dokumente anhand der Avro-Schema-Spezifikation.

```
raptorxml valavroschema | avroschema [options] AvroSchema
```

- Das Argument *AvroSchema* ist das zu validierende Avro-Schema-Dokument.
- Um mehrere Arvo-Schemas zu validieren, (i) listen Sie entweder die zu validierenden Dateien im CLI auf, wobei jede Datei durch ein Leerzeichen von der nächsten getrennt wird, oder (ii) listen Sie die zu validierenden Dateien in einer Textdatei (`.txt`-Datei) auf (ein Dateiname pro Zeile) und geben Sie diese

Textdatei zusammen mit der auf `true` gesetzten Option `--listfile` 252 (*siehe Liste der Optionen unten*) als das *AvroSchema*-Argument an.

Beispiele

Beispiele für den Befehl **valavroschema**:

- **raptorxml** valavroschema c:\MyAvroSchema.avsc
- **raptorxml** valavroschema c:\MyAvroSchema01.avsc c:\MyAvroSchema02.avsc
- **raptorxml** avroschema --listfile=true c:\MyFileList.txt

▼ Groß- und Kleinschreibung und Schrägstriche in der Befehlszeile

RaptorXML (und **RaptorXMLServer** für Administrator-Befehle) *unter Windows*
raptorxml und **raptorxmlserver** für Administrator-Befehle) *unter Windows und Unix (Linux, Mac)*

* Beachten Sie, dass klein geschriebene Befehle (`raptorxml` und `raptorxmlserver`) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (`RaptorXML`) nur unter Windows und Mac ausgeführt werden.
*Verwenden Sie auf Linux und Mac-Systemen Schrägstriche und auf Windows-Systemen umgekehrte Schrägstriche.

▼ Umgekehrte Schrägstriche, Leerzeichen und Sonderzeichen auf Windows-Systemen

Auf Windows-Systemen: Verwenden Sie bei Vorhandensein von Leerzeichen oder Sonerzeichen in Strings (z.B. in Datei- oder Ordnernamen oder Firmen-, Personen- oder Produktnamen) Anführungszeichen: z.B., **"Meine Datei"**. Beachten Sie jedoch, dass ein von einem doppelten Anführungszeichen gefolgter umgekehrter Schrägstrich (z.B: **"C:\Mein Verzeichnis\"**) eventuell nicht korrekt gelesen wird, da der umgekehrte Schrägstrich auch den Beginn einer Escape-Sequenz markiert und die Escape-Sequenz **\"** für ein doppeltes Anführungszeichen steht. Wenn Sie diese Zeichensequenz mit einem Escape versehen wollen, verwenden Sie einen vorangestellten umgekehrten Schrägstrich, wie den folgenden: **\\"**. Einfacher ausgedrückt: Wenn Sie einen Dateipfad, der Leerzeichen oder einen umgekehrten Schrägstrich am Ende enthält, schreiben müssen, so schreiben Sie diesen folgendermaßen: **"C:\Mein Verzeichnis\\"**.

Optionen

Optionen werden in ihrer kurzen Form (falls verfügbar) und in ihrer langen Form aufgelistet. Für die kurze und die lange Form können ein oder zwei Bindestriche verwendet werden. Eine Option kann, muss aber keinen Wert erhalten. Eine Option, die einen Wert erhält, wird folgendermaßen geschrieben: **--option=wert**. Werte können außer in zwei Fällen ohne Anführungszeichen definiert werden: (i) wenn der Wertestring Leerzeichen enthält oder (ii) wenn in der Beschreibung der Option explizit erwähnt ist, dass Anführungszeichen zwingend erforderlich sind. Wenn eine Option einen Booleschen Wert erhält und kein Wert definiert ist, so ist der Standardwert der Option `TRUE`. Mit Hilfe der Option **--h, --help** können Sie Informationen über den Befehl anzeigen.

▼ Verarbeitung

▼ listfile

--listfile = `true|false`
Bei `true` wird das Argument *InputFile* des Befehls als Textdatei behandelt, die einen Dateinamen pro Zeile enthält. Der Standardwert ist `false`. (Als Alternative können die Dateien im CLI getrennt

durch ein Leerzeichen aufgelistet werden. Beachten Sie allerdings, dass CLIs eine maximale Zeichenanzahl haben). Beachten Sie, dass die Option `--listfile` nur auf Argumente, nicht aber auf Optionen angewendet wird.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- recurse

  `--recurse =` `true|false`
  Dient zur Auswahl von Dateien innerhalb von Unterverzeichnissen einschließlich ZIP-Archiven. Bei `true` wählt das Argument *`InputFile`* des Befehls die angegebene Datei auch in den Unterverzeichnissen aus. Beispiel: `"test.zip|zip\test.xml"` wählt Dateien mit dem Namen `test.xml` auf allen Ordnerebenen des ZIP-Ordners aus. Referenzen auf ZIP-Dateien müssen in Anführungszeichen angegeben werden. Es können die Platzhalter `*` und `?` verwendet werden. Mit `*.xml` werden folglich alle `.xml` Dateien im (ZIP-)Ordner ausgewählt. Der Standardwert der Option ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- Kataloge und globale Ressourcen
  - catalog

    `--catalog =` *`FILE`*
    Gibt den absoluten Pfad zu einer Root-Katalog-Datei an, die nicht die installierte Root-Katalog-Datei ist. Der Standardwert ist der absolute Pfad zur installierten Root-Katalog-Datei. (`<installationsordner>\Altova\RaptorXMLServer2025\etc\RootCatalog.xml`). Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge[48].

  - user-catalog

    `--user-catalog =` *`FILE`*
    Definiert den absoluten Pfad zu einem XML-Katalog, der zusätzlich zum Root-Katalog verwendet werden soll. Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge[48].

  - enable-globalresources

    `--enable-globalresources =` `true|false`
    Aktiviert die globalen Ressourcen[55]. Standardwert ist `false`.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

  - globalresourceconfig [gc]

    `--gc | --globalresourceconfig =` *`VALUE`*
    Definiert die aktive Konfiguration der globalen Ressource[55] (und aktiviert globale Ressourcen)[55].

  - globalresourcefile [gr]

    `--gr | --globalresourcefile =` *`FILE`*
    Definiert die globale Ressourcendatei[55] (und aktiviert globale Ressourcen)[55].

- Allgemeine Optionen
  - error-format

    `--error-format =` `text|shortxml|longxml`
    Definiert das Format der Fehlerausgabe. Der Standardwert ist `text`. Mit den anderen Optionen werden XML-Formate generiert, wobei mit `longxml` mehr Details generiert werden.

  - error-limit

    `--error-limit =` *`N | unlimited`*
    Definiert das Fehlerlimit mit einem Wertebereich von `1` bis `9999` oder `unbegrenzt`. Der Standardwert ist `100`. Bei Erreichung des Fehlerlimits wird die Validierung gestoppt. Dient dazu, die Prozessorverwendung während der Validierung/Transformation einzuschränken.

  - info-limit

    `--info-limit =` *`N | unlimited`*
    Definiert die Grenze für Informationsmeldungen im Bereich von `1-65535` oder als `unlimited` (unbegrenzt). Die Verarbeitung wird auch nach Erreichen des Info-Limits fortgesetzt, doch werden weitere Meldungen nicht mehr ausgegeben. Der Standardwert ist `100`.

  - help

    `--help`
    Zeigt den Hilfetext zum Befehl an. Beispiel: `valany --h`. (Alternativ dazu kann der Befehl `help` zusammen mit einem Argument verwendet werden. Beispiel: `help valany`.)

  - listfile

    `--listfile =` `true|false`
    Bei `true` wird das Argument *`InputFile`* des Befehls als Textdatei behandelt, die einen Dateinamen pro Zeile enthält. Der Standardwert ist `false`. (Als Alternative können die Dateien im CLI getrennt durch ein Leerzeichen aufgelistet werden. Beachten Sie allerdings, dass CLIs eine maximale Zeichenanzahl haben). Beachten Sie, dass die Option `--listfile` nur auf Argumente, nicht aber auf Optionen angewendet wird.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

  - log-output

    `--log-output =` *`FILE`*
    Schreibt die Meldungsausgabe in die angegebene URL. Stellen Sie sicher, dass das CLI Schreibrechte für den Ausgabepfad hat.

  - network-timeout

    `--network-timeout =` *`VALUE`*
    Definiert das Timeout für entfernte I/O-Operationen in Millisekunden. Der Standardwert ist: `40000`.

  - recurse

    `--recurse =` `true|false`

Dient zur Auswahl von Dateien innerhalb von Unterverzeichnissen einschließlich ZIP-Archiven. Bei `true` wählt das Argument *InputFile* des Befehls die angegebene Datei auch in den Unterverzeichnissen aus. Beispiel: `"test.zip|zip\test.xml"` wählt Dateien mit dem Namen `test.xml` auf allen Ordnerebenen des ZIP-Ordners aus. Referenzen auf ZIP-Dateien müssen in Anführungszeichen angegeben werden. Es können die Platzhalter `*` und `?` verwendet werden. Mit `*.xml` werden folglich alle `.xml` Dateien im (ZIP-)Ordner ausgewählt. Der Standardwert der Option ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- verbose

  `--verbose =` `true|false`
  Mit dem Wert `true` wird die Ausgabe zusätzlicher Informationen bei der Validierung aktiviert. Der Standardwert ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- verbose-output

  `--verbose-output =` *FILE*
  Schreibt die ausführliche Ausgabe in *FILE*.

- version

  `--version`
  Zeigt die Version von RaptorXML Server an. Setzen Sie `--version` bei Verwendung mit einem Befehl vor den Befehl.

- warning-limit

  `--warning-limit =` *N | unlimited*
  Definiert das Warnungslimit im Bereich von `1-65535` oder `unbegrenzt`. Bei Erreichen dieses Limits wird die Verarbeitung fortgesetzt, doch werden keine weiteren Warnungen mehr ausgegeben. Der Standardwert ist `100`.

5.5.7 valjsonschema (jsonschema)

Der Befehl **`valjsonschema | jsonschema`** validiert ein oder mehrere JSON-Schema-Dokumente anhand der (mit der Option **`jsonschema-version`** definierten) JSON Schema-Spezifikationen.

```
raptorxml valjsonschema | jsonschema [options] InputFile
```

- Beim Argument *InputFile* handelt es sich um das zu validierende JSON-Schema-Dokument.
- Um mehrere Dokumente zu validieren, (i) listen Sie entweder die zu validierenden Dateien im CLI auf, wobei jede Datei durch ein Leerzeichen von der nächsten getrennt wird, oder (ii) listen Sie die zu validierenden Dateien in einer Textdatei (`.txt`-Datei) auf (ein Dateiname pro Zeile) und geben Sie diese

Textdatei zusammen mit der auf `true` gesetzten Option `--listfile` 252 als das *`InputFile`* Argument an (*siehe Optionsliste unten*).

Beispiele

Beispiele für den Befehl **`valjsonschema`**:

- **`raptorxml`** `valjsonschema c:\MyJSONSchema.json`
- **`raptorxml`** `jsonschema c:\MyJSONSchema-01.json c:\MyJSONSchema-02.json`
- **`raptorxml`** `jsonschema --listfile=true c:\FileList.txt`

▼ Groß- und Kleinschreibung und Schrägstriche in der Befehlszeile

`RaptorXML` (und **`RaptorXMLServer`** für Administrator-Befehle) *unter Windows*
`raptorxml` und **`raptorxmlserver`** für Administrator-Befehle) *unter Windows und Unix (Linux, Mac)*

* Beachten Sie, dass klein geschriebene Befehle (`raptorxml` und `raptorxmlserver`) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (`RaptorXML`) nur unter Windows und Mac ausgeführt werden.
*Verwenden Sie auf Linux und Mac-Systemen Schrägstriche und auf Windows-Systemen umgekehrte Schrägstriche.

▼ Umgekehrte Schrägstriche, Leerzeichen und Sonderzeichen auf Windows-Systemen

Auf Windows-Systemen: Verwenden Sie bei Vorhandensein von Leerzeichen oder Sonerzeichen in Strings (z.B. in Datei- oder Ordnernamen oder Firmen-, Personen- oder Produktnamen) Anführungszeichen: z.B., **`"Meine Datei"`**. Beachten Sie jedoch, dass ein von einem doppelten Anführungszeichen gefolgter umgekehrter Schrägstrich (z.B: **`"C:\Mein Verzeichnis\"`**) eventuell nicht korrekt gelesen wird, da der umgekehrte Schrägstrich auch den Beginn einer Escape-Sequenz markiert und die Escape-Sequenz **`\"`** für ein doppeltes Anführungszeichen steht. Wenn Sie diese Zeichensequenz mit einem Escape versehen wollen, verwenden Sie einen vorangestellten umgekehrten Schrägstrich, wie den folgenden: **`\\"`**. Einfacher ausgedrückt: Wenn Sie einen Dateipfad, der Leerzeichen oder einen umgekehrten Schrägstrich am Ende enthält, schreiben müssen, so schreiben Sie diesen folgendermaßen: **`"C:\Mein Verzeichnis\\"`**.

Optionen

Optionen werden in ihrer kurzen Form (falls verfügbar) und in ihrer langen Form aufgelistet. Für die kurze und die lange Form können ein oder zwei Bindestriche verwendet werden. Eine Option kann, muss aber keinen Wert erhalten. Eine Option, die einen Wert erhält, wird folgendermaßen geschrieben: **`--option=wert`**. Werte können außer in zwei Fällen ohne Anführungszeichen definiert werden: (i) wenn der Wertestring Leerzeichen enthält oder (ii) wenn in der Beschreibung der Option explizit erwähnt ist, dass Anführungszeichen zwingend erforderlich sind. Wenn eine Option einen Booleschen Wert erhält und kein Wert definiert ist, so ist der Standardwert der Option `TRUE`. Mit Hilfe der Option **`--h, --help`** können Sie Informationen über den Befehl anzeigen.

▼ Validierung und Verarbeitung

▼ listfile

`--listfile =` `true|false`
Bei `true` wird das Argument *`InputFile`* des Befehls als Textdatei behandelt, die einen Dateinamen pro Zeile enthält. Der Standardwert ist `false`. (Als Alternative können die Dateien im CLI getrennt

durch ein Leerzeichen aufgelistet werden. Beachten Sie allerdings, dass CLIs eine maximale Zeichenanzahl haben). Beachten Sie, dass die Option `--listfile` nur auf Argumente, nicht aber auf Optionen angewendet wird.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- recurse

  `--recurse =` `true|false`
  Dient zur Auswahl von Dateien innerhalb von Unterverzeichnissen einschließlich ZIP-Archiven. Bei `true` wählt das Argument *`InputFile`* des Befehls die angegebene Datei auch in den Unterverzeichnissen aus. Beispiel: `"test.zip|zip\test.xml"` wählt Dateien mit dem Namen `test.xml` auf allen Ordnerebenen des ZIP-Ordners aus. Referenzen auf ZIP-Dateien müssen in Anführungszeichen angegeben werden. Es können die Platzhalter `*` und `?` verwendet werden. Mit `*.xml` werden folglich alle `.xml` Dateien im (ZIP-)Ordner ausgewählt. Der Standardwert der Option ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- JSON-Validierungsoptionen
  - additional-schema

    `--additional-schema =` *`FILE`*
    Definiert URIs eines zusätzlichen Schema-Dokuments. Das zusätzliche Schema wird vom Hauptschema geladen und kann vom Hauptschema aus über die Eigenschaft **`id`** oder **`$id`** des zusätzlichen Schemas referenziert werden.

  - disable-format-checks

    `--disable-format-checks =` `true|false`
    Deaktiviert die durch das format-Attribut erzwungene semantische Validierung. Der Standardwert ist `false`.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

  - jsonschema-format

    `--jsonschema-format =` `json|yaml`
    Definiert das Format, in dem das JSON-Schema geschrieben ist: JSON oder YAML. Der Standardwert ist `json`.

  - jsonschema-version

    `--jsonschema-version =` `draft04|draft06|draft07|2019-09|2020-12|oas-3.1|latest|detect`
    Definiert, welche Version der JSON-Schema-Draft-Spezifikation verwendet werden soll. Der Standardwert ist `detect`.

  - strict-integer-checks

    `--strict-integer-checks =` `true|false`

Gibt an, ob bei neueren Schemas, in denen die Ganzzahlüberprüfung weniger streng gehandhabt wird, die strengere Ganzzahlüberprüfung aus draft-04 verwendet werden soll. So ist z.B. `1.0` in draft-04 keine gültige Ganzzahl, in späteren Drafts jedoch schon. Diese Option hat keine Auswirkung auf draft-04-Schemas. Der Standardwert der Option ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- Kataloge und globale Ressourcen
  - catalog

    `--catalog =` *FILE*
    Gibt den absoluten Pfad zu einer Root-Katalog-Datei an, die nicht die installierte Root-Katalog-Datei ist. Der Standardwert ist der absolute Pfad zur installierten Root-Katalog-Datei. (`<installationsordner>\Altova\RaptorXMLServer2025\etc\RootCatalog.xml`). Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge[48].

  - user-catalog

    `--user-catalog =` *FILE*
    Definiert den absoluten Pfad zu einem XML-Katalog, der zusätzlich zum Root-Katalog verwendet werden soll. Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge[48].

  - enable-globalresources

    `--enable-globalresources =` `true|false`
    Aktiviert die globalen Ressourcen[55]. Standardwert ist `false`.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

  - globalresourceconfig [gc]

    `--gc | --globalresourceconfig =` *VALUE*
    Definiert die aktive Konfiguration der globalen Ressource[55] (und aktiviert globale Ressourcen)[55].

  - globalresourcefile [gr]

    `--gr | --globalresourcefile =` *FILE*
    Definiert die globale Ressourcendatei[55] (und aktiviert globale Ressourcen)[55].

- Allgemeine Optionen
  - error-format

    `--error-format =` `text|shortxml|longxml`
    Definiert das Format der Fehlerausgabe. Der Standardwert ist `text`. Mit den anderen Optionen werden XML-Formate generiert, wobei mit `longxml` mehr Details generiert werden.

  - error-limit

    `--error-limit =` *N | unlimited*

Definiert das Fehlerlimit mit einem Wertebereich von `1` bis `9999` oder `unbegrenzt`. Der Standardwert ist `100`. Bei Erreichung des Fehlerlimits wird die Validierung gestoppt. Dient dazu, die Prozessorverwendung während der Validierung/Transformation einzuschränken.

- info-limit

  `--info-limit =` *`N | unlimited`*
  Definiert die Grenze für Informationsmeldungen im Bereich von `1-65535` oder als `unlimited` (unbegrenzt). Die Verarbeitung wird auch nach Erreichen des Info-Limits fortgesetzt, doch werden weitere Meldungen nicht mehr ausgegeben. Der Standardwert ist `100`.

- help

  `--help`
  Zeigt den Hilfetext zum Befehl an. Beispiel: `valany --h`. (Alternativ dazu kann der Befehl `help` zusammen mit einem Argument verwendet werden. Beispiel: `help valany`.)

- listfile

  `--listfile =` `true|false`
  Bei `true` wird das Argument *`InputFile`* des Befehls als Textdatei behandelt, die einen Dateinamen pro Zeile enthält. Der Standardwert ist `false`. (Als Alternative können die Dateien im CLI getrennt durch ein Leerzeichen aufgelistet werden. Beachten Sie allerdings, dass CLIs eine maximale Zeichenanzahl haben). Beachten Sie, dass die Option `--listfile` nur auf Argumente, nicht aber auf Optionen angewendet wird.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- log-output

  `--log-output =` *`FILE`*
  Schreibt die Meldungsausgabe in die angegebene URL. Stellen Sie sicher, dass das CLI Schreibrechte für den Ausgabepfad hat.

- network-timeout

  `--network-timeout =` *`VALUE`*
  Definiert das Timeout für entfernte I/O-Operationen in Millisekunden. Der Standardwert ist: `40000`.

- recurse

  `--recurse =` `true|false`
  Dient zur Auswahl von Dateien innerhalb von Unterverzeichnissen einschließlich ZIP-Archiven. Bei `true` wählt das Argument *`InputFile`* des Befehls die angegebene Datei auch in den Unterverzeichnissen aus. Beispiel: `"test.zip|zip\test.xml"` wählt Dateien mit dem Namen `test.xml` auf allen Ordnerebenen des ZIP-Ordners aus. Referenzen auf ZIP-Dateien müssen in Anführungszeichen angegeben werden. Es können die Platzhalter `*` und `?` verwendet werden. Mit `*.xml` werden folglich alle `.xml` Dateien im (ZIP-)Ordner ausgewählt. Der Standardwert der Option ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- verbose

--verbose = true|false
Mit dem Wert true wird die Ausgabe zusätzlicher Informationen bei der Validierung aktiviert. Der Standardwert ist false.
Hinweis: Die Booleschen Optionswerte werden auf true gesetzt, wenn die Option ohne einen Wert definiert wird.

- verbose-output

  --verbose-output = *FILE*
  Schreibt die ausführliche Ausgabe in *FILE*.

- version

  --version
  Zeigt die Version von RaptorXML Server an. Setzen Sie --version bei Verwendung mit einem Befehl vor den Befehl.

- warning-limit

  --warning-limit = *N | unlimited*
  Definiert das Warnungslimit im Bereich von 1-65535 oder unbegrenzt. Bei Erreichen dieses Limits wird die Verarbeitung fortgesetzt, doch werden keine weiteren Warnungen mehr ausgegeben. Der Standardwert ist 100.

5.5.8 valjson (json)

Der Befehl **valjson | json** validiert ein oder mehrere JSON-Instanzdokumente anhand des mit der Option --schema (--jsonschema) bereitgestellten JSON-Schemas.

```
raptorxml valjson | json [options] --jsonschema=File InputFile
```

- Das Argument *InputFile* ist das zu validierende JSON-Instanzdokument.
- Um mehrere Dokumente zu validieren, (i) listen Sie entweder die zu validierenden Dateien im CLI auf, wobei jede Datei durch ein Leerzeichen von der nächsten getrennt wird, oder (ii) listen Sie die zu validierenden Dateien in einer Textdatei (.txt-Datei) auf (ein Dateiname pro Zeile) und geben Sie diese Textdatei zusammen mit der auf true gesetzten Option --listfile[252] als das *InputFile* Argument an (*siehe Optionsliste unten*).

Beispiele

Beispiele für den Befehl **valjson**:

- **raptorxml** valjson --jsonschema=c:\MyJSONSchema.json c:\MyJSONInstance.json
- **raptorxml** json --jsonschema=c:\MyJSONSchema.json c:\MyJSONInstance-01.json c:\MyJSONInstance-02.json
- **raptorxml** json --jsonschema=c:\MyJSONSchema.json --listfile=true c:\FileList.txt

- Groß- und Kleinschreibung und Schrägstriche in der Befehlszeile

  RaptorXML (und **RaptorXMLServer** für Administrator-Befehle) *unter Windows*
  raptorxml und **raptorxmlserver** für Administrator-Befehle) *unter Windows und Unix (Linux, Mac)*

  * Beachten Sie, dass klein geschriebene Befehle (raptorxml und raptorxmlserver) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (RaptorXML) nur unter Windows und Mac ausgeführt werden.
  *Verwenden Sie auf Linux und Mac-Systemen Schrägstriche und auf Windows-Systemen umgekehrte Schrägstriche.

- Umgekehrte Schrägstriche, Leerzeichen und Sonderzeichen auf Windows-Systemen

  Auf Windows-Systemen: Verwenden Sie bei Vorhandensein von Leerzeichen oder Sonerzeichen in Strings (z.B. in Datei- oder Ordnernamen oder Firmen-, Personen- oder Produktnamen) Anführungszeichen: z.B., **"Meine Datei"**. Beachten Sie jedoch, dass ein von einem doppelten Anführungszeichen gefolgter umgekehrter Schrägstrich (z.B: **"C:\Mein Verzeichnis\"**) eventuell nicht korrekt gelesen wird, da der umgekehrte Schrägstrich auch den Beginn einer Escape-Sequenz markiert und die Escape-Sequenz **\"** für ein doppeltes Anführungszeichen steht. Wenn Sie diese Zeichensequenz mit einem Escape versehen wollen, verwenden Sie einen vorangestellten umgekehrten Schrägstrich, wie den folgenden: **\\"**. Einfacher ausgedrückt: Wenn Sie einen Dateipfad, der Leerzeichen oder einen umgekehrten Schrägstrich am Ende enthält, schreiben müssen, so schreiben Sie diesen folgendermaßen: **"C:\Mein Verzeichnis\\"**.

Optionen

Optionen werden in ihrer kurzen Form (falls verfügbar) und in ihrer langen Form aufgelistet. Für die kurze und die lange Form können ein oder zwei Bindestriche verwendet werden. Eine Option kann, muss aber keinen Wert erhalten. Eine Option, die einen Wert erhält, wird folgendermaßen geschrieben: **--option=wert**. Werte können außer in zwei Fällen ohne Anführungszeichen definiert werden: (i) wenn der Wertestring Leerzeichen enthält oder (ii) wenn in der Beschreibung der Option explizit erwähnt ist, dass Anführungszeichen zwingend erforderlich sind. Wenn eine Option einen Booleschen Wert erhält und kein Wert definiert ist, so ist der Standardwert der Option TRUE. Mit Hilfe der Option **--h, --help** können Sie Informationen über den Befehl anzeigen.

- Validierung und Verarbeitung
  - schema, jsonschema

    --schema = *FILE*, **--jsonschema =** *FILE*
    Definiert den Pfad zu dem JSON-Schema-Dokument, anhand dessen die JSON-Instanzdokumente validiert werden sollen.

  - jsonschema-format

    --jsonschema-format = json|yaml
    Definiert das Format, in dem das JSON-Schema geschrieben ist: JSON oder YAML. Der Standardwert ist json.

  - jsonschema-version

    --jsonschema-version = draft04|draft06|draft07|2019-09|2020-12|oas-3.1|latest|detect
    Definiert, welche Version der JSON-Schema-Draft-Spezifikation verwendet werden soll. Der

Standardwert ist `detect`.

- listfile

  `--listfile =` `true|false`
  Bei `true` wird das Argument *`InputFile`* des Befehls als Textdatei behandelt, die einen Dateinamen pro Zeile enthält. Der Standardwert ist `false`. (Als Alternative können die Dateien im CLI getrennt durch ein Leerzeichen aufgelistet werden. Beachten Sie allerdings, dass CLIs eine maximale Zeichenanzahl haben). Beachten Sie, dass die Option `--listfile` nur auf Argumente, nicht aber auf Optionen angewendet wird.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- additional-schema

  `--additional-schema =` *`FILE`*
  Definiert URIs eines zusätzlichen Schema-Dokuments. Das zusätzliche Schema wird vom Hauptschema geladen und kann vom Hauptschema aus über die Eigenschaft **`id`** oder **`$id`** des zusätzlichen Schemas referenziert werden.

- recurse

  `--recurse =` `true|false`
  Dient zur Auswahl von Dateien innerhalb von Unterverzeichnissen einschließlich ZIP-Archiven. Bei `true` wählt das Argument *`InputFile`* des Befehls die angegebene Datei auch in den Unterverzeichnissen aus. Beispiel: `"test.zip|zip\test.xml"` wählt Dateien mit dem Namen `test.xml` auf allen Ordnerebenen des ZIP-Ordners aus. Referenzen auf ZIP-Dateien müssen in Anführungszeichen angegeben werden. Es können die Platzhalter `*` und `?` verwendet werden. Mit `*.xml` werden folglich alle `.xml` Dateien im (ZIP-)Ordner ausgewählt. Der Standardwert der Option ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- json5

  `--json5 =` `true|false`
  Aktiviert die JSON5-Unterstützung. Der Standardwert ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- jsonc

  `--jsonc =` `true|false`
  Aktiviert die Unterstützung für Kommentare in JSON-Dokumenten. Der Standardwert ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- json-lines

  `--json-lines =` `true|false`
  Aktiviert die Unterstützung für JSON Lines (d.h. einen JSON-Wert pro Zeile). Der Standardwert ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert

definiert wird.

- disable-format-checks

  --disable-format-checks = `true|false`
  Deaktiviert die durch das format-Attribut erzwungene semantische Validierung. Der Standardwert ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- strict-integer-checks

  --strict-integer-checks = `true|false`
  Gibt an, ob bei neueren Schemas, in denen die Ganzzahlüberprüfung weniger streng gehandhabt wird, die strengere Ganzzahlüberprüfung aus draft-04 verwendet werden soll. So ist z.B. `1.0` in draft-04 keine gültige Ganzzahl, in späteren Drafts jedoch schon. Diese Option hat keine Auswirkung auf draft-04-Schemas. Der Standardwert der Option ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- Kataloge und globale Ressourcen
  - catalog

    --catalog = *FILE*
    Gibt den absoluten Pfad zu einer Root-Katalog-Datei an, die nicht die installierte Root-Katalog-Datei ist. Der Standardwert ist der absolute Pfad zur installierten Root-Katalog-Datei. (`<installationsordner>\Altova\RaptorXMLServer2025\etc\RootCatalog.xml`). Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge 48.

  - user-catalog

    --user-catalog = *FILE*
    Definiert den absoluten Pfad zu einem XML-Katalog, der zusätzlich zum Root-Katalog verwendet werden soll. Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge 48.

  - enable-globalresources

    --enable-globalresources = `true|false`
    Aktiviert die globalen Ressourcen 55. Standardwert ist `false`.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

  - globalresourceconfig [gc]

    --gc | --globalresourceconfig = *VALUE*
    Definiert die aktive Konfiguration der globalen Ressource 55 (und aktiviert globale Ressourcen) 55.

  - globalresourcefile [gr]

    --gr | --globalresourcefile = *FILE*
    Definiert die globale Ressourcendatei 55 (und aktiviert globale Ressourcen) 55.

- Allgemeine Optionen
  - error-format

    `--error-format =` `text|shortxml|longxml`
    Definiert das Format der Fehlerausgabe. Der Standardwert ist `text`. Mit den anderen Optionen werden XML-Formate generiert, wobei mit `longxml` mehr Details generiert werden.

  - error-limit

    `--error-limit =` *`N | unlimited`*
    Definiert das Fehlerlimit mit einem Wertebereich von `1` bis `9999` oder `unbegrenzt`. Der Standardwert ist `100`. Bei Erreichung des Fehlerlimits wird die Validierung gestoppt. Dient dazu, die Prozessorverwendung während der Validierung/Transformation einzuschränken.

  - info-limit

    `--info-limit =` *`N | unlimited`*
    Definiert die Grenze für Informationsmeldungen im Bereich von `1-65535` oder als `unlimited` (unbegrenzt). Die Verarbeitung wird auch nach Erreichen des Info-Limits fortgesetzt, doch werden weitere Meldungen nicht mehr ausgegeben. Der Standardwert ist `100`.

  - help

    `--help`
    Zeigt den Hilfetext zum Befehl an. Beispiel: `valany --h`. (Alternativ dazu kann der Befehl `help` zusammen mit einem Argument verwendet werden. Beispiel: `help valany`.)

  - listfile

    `--listfile =` `true|false`
    Bei `true` wird das Argument *`InputFile`* des Befehls als Textdatei behandelt, die einen Dateinamen pro Zeile enthält. Der Standardwert ist `false`. (Als Alternative können die Dateien im CLI getrennt durch ein Leerzeichen aufgelistet werden. Beachten Sie allerdings, dass CLIs eine maximale Zeichenanzahl haben). Beachten Sie, dass die Option `--listfile` nur auf Argumente, nicht aber auf Optionen angewendet wird.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

  - log-output

    `--log-output =` *`FILE`*
    Schreibt die Meldungsausgabe in die angegebene URL. Stellen Sie sicher, dass das CLI Schreibrechte für den Ausgabepfad hat.

  - network-timeout

    `--network-timeout =` *`VALUE`*
    Definiert das Timeout für entfernte I/O-Operationen in Millisekunden. Der Standardwert ist: `40000`.

  - recurse

    `--recurse =` `true|false`

Dient zur Auswahl von Dateien innerhalb von Unterverzeichnissen einschließlich ZIP-Archiven. Bei `true` wählt das Argument *`InputFile`* des Befehls die angegebene Datei auch in den Unterverzeichnissen aus. Beispiel: `"test.zip|zip\test.xml"` wählt Dateien mit dem Namen `test.xml` auf allen Ordnerebenen des ZIP-Ordners aus. Referenzen auf ZIP-Dateien müssen in Anführungszeichen angegeben werden. Es können die Platzhalter `*` und `?` verwendet werden. Mit `*.xml` werden folglich alle `.xml` Dateien im (ZIP-)Ordner ausgewählt. Der Standardwert der Option ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- verbose

  `--verbose =` `true|false`
  Mit dem Wert `true` wird die Ausgabe zusätzlicher Informationen bei der Validierung aktiviert. Der Standardwert ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- verbose-output

  `--verbose-output =` *`FILE`*
  Schreibt die ausführliche Ausgabe in *`FILE`*.

- version

  `--version`
  Zeigt die Version von RaptorXML Server an. Setzen Sie `--version` bei Verwendung mit einem Befehl vor den Befehl.

- warning-limit

  `--warning-limit =` *`N | unlimited`*
  Definiert das Warnungslimit im Bereich von `1-65535` oder `unbegrenzt`. Bei Erreichen dieses Limits wird die Verarbeitung fortgesetzt, doch werden keine weiteren Warnungen mehr ausgegeben. Der Standardwert ist `100`.

5.5.9 valyaml (yaml)

Der Befehl **`valyaml | yaml`** validiert ein oder mehrere YAML-Instanzdokumente anhand des mit der Option `--schema` (`--jsonschema`) bereitgestellten JSON-Schemas.

```
raptorxml valyaml | yaml [options] --jsonschema=File InputFile
```

- Das Argument *`InputFile`* ist das zu validierende YAML-Instanzdokument.
- Um mehrere Dokumente zu validieren, (i) listen Sie entweder die zu validierenden Dateien im CLI auf, wobei jede Datei durch ein Leerzeichen von der nächsten getrennt wird, oder (ii) listen Sie die zu validierenden Dateien in einer Textdatei (`.txt`-Datei) auf (ein Dateiname pro Zeile) und geben Sie diese

Textdatei zusammen mit der auf `true` gesetzten Option `--listfile` 252 als das *`InputFile`* Argument an (*siehe Optionsliste unten*).

Beispiele

Beispiele für den Befehl **`valyaml`**:

- **`raptorxml`** `valyaml --jsonschema=c:\MyJSONSchema.json c:\MyYAMLInstance.yaml`
- **`raptorxml`** `yaml --jsonschema=c:\MyJSONSchema.json c:\MyYAMLInstance-01.yaml c:\MyYAMLInstance-02.yaml`
- **`raptorxml`** `yaml --jsonschema=c:\MyJSONSchema.json --listfile=true c:\FileList.txt`

▼ Groß- und Kleinschreibung und Schrägstriche in der Befehlszeile

`RaptorXML` (und **`RaptorXMLServer`** für Administrator-Befehle) *unter Windows*
`raptorxml` und **`raptorxmlserver`** für Administrator-Befehle) *unter Windows und Unix (Linux, Mac)*

* Beachten Sie, dass klein geschriebene Befehle (`raptorxml` und `raptorxmlserver`) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (`RaptorXML`) nur unter Windows und Mac ausgeführt werden.
*Verwenden Sie auf Linux und Mac-Systemen Schrägstriche und auf Windows-Systemen umgekehrte Schrägstriche.

▼ Umgekehrte Schrägstriche, Leerzeichen und Sonderzeichen auf Windows-Systemen

Auf Windows-Systemen: Verwenden Sie bei Vorhandensein von Leerzeichen oder Sonerzeichen in Strings (z.B. in Datei- oder Ordnernamen oder Firmen-, Personen- oder Produktnamen) Anführungszeichen: z.B., **`"Meine Datei"`**. Beachten Sie jedoch, dass ein von einem doppelten Anführungszeichen gefolgter umgekehrter Schrägstrich (z.B: **`"C:\Mein Verzeichnis\"`**) eventuell nicht korrekt gelesen wird, da der umgekehrte Schrägstrich auch den Beginn einer Escape-Sequenz markiert und die Escape-Sequenz **`\"`** für ein doppeltes Anführungszeichen steht. Wenn Sie diese Zeichensequenz mit einem Escape versehen wollen, verwenden Sie einen vorangestellten umgekehrten Schrägstrich, wie den folgenden: **`\\"`**. Einfacher ausgedrückt: Wenn Sie einen Dateipfad, der Leerzeichen oder einen umgekehrten Schrägstrich am Ende enthält, schreiben müssen, so schreiben Sie diesen folgendermaßen: **`"C:\Mein Verzeichnis\\"`**.

Optionen

Optionen werden in ihrer kurzen Form (falls verfügbar) und in ihrer langen Form aufgelistet. Für die kurze und die lange Form können ein oder zwei Bindestriche verwendet werden. Eine Option kann, muss aber keinen Wert erhalten. Eine Option, die einen Wert erhält, wird folgendermaßen geschrieben: **`--option=wert`**. Werte können außer in zwei Fällen ohne Anführungszeichen definiert werden: (i) wenn der Wertestring Leerzeichen enthält oder (ii) wenn in der Beschreibung der Option explizit erwähnt ist, dass Anführungszeichen zwingend erforderlich sind. Wenn eine Option einen Booleschen Wert erhält und kein Wert definiert ist, so ist der Standardwert der Option `TRUE`. Mit Hilfe der Option **`--h, --help`** können Sie Informationen über den Befehl anzeigen.

▼ Validierung und Verarbeitung

▼ schema, jsonschema

`--schema =` *`FILE`*, **`--jsonschema =`** *`FILE`*
Definiert den Pfad zu dem JSON-Schema-Dokument, anhand dessen die JSON-Instanzdokumente

validiert werden sollen.

- jsonschema-format

  --jsonschema-format = `json|yaml`
  Definiert das Format, in dem das JSON-Schema geschrieben ist: JSON oder YAML. Der Standardwert ist `json`.

- jsonschema-version

  --jsonschema-version = `draft04|draft06|draft07|2019-09|2020-12|oas-3.1|latest|detect`
  Definiert, welche Version der JSON-Schema-Draft-Spezifikation verwendet werden soll. Der Standardwert ist `detect`.

- listfile

  --listfile = `true|false`
  Bei `true` wird das Argument *`InputFile`* des Befehls als Textdatei behandelt, die einen Dateinamen pro Zeile enthält. Der Standardwert ist `false`. (Als Alternative können die Dateien im CLI getrennt durch ein Leerzeichen aufgelistet werden. Beachten Sie allerdings, dass CLIs eine maximale Zeichenanzahl haben). Beachten Sie, dass die Option `--listfile` nur auf Argumente, nicht aber auf Optionen angewendet wird.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- additional-schema

  --additional-schema = *`FILE`*
  Definiert URIs eines zusätzlichen Schema-Dokuments. Das zusätzliche Schema wird vom Hauptschema geladen und kann vom Hauptschema aus über die Eigenschaft **`id`** oder **`$id`** des zusätzlichen Schemas referenziert werden.

- recurse

  --recurse = `true|false`
  Dient zur Auswahl von Dateien innerhalb von Unterverzeichnissen einschließlich ZIP-Archiven. Bei `true` wählt das Argument *`InputFile`* des Befehls die angegebene Datei auch in den Unterverzeichnissen aus. Beispiel: `"test.zip|zip\test.xml"` wählt Dateien mit dem Namen `test.xml` auf allen Ordnerebenen des ZIP-Ordners aus. Referenzen auf ZIP-Dateien müssen in Anführungszeichen angegeben werden. Es können die Platzhalter `*` und `?` verwendet werden. Mit `*.xml` werden folglich alle `.xml` Dateien im (ZIP-)Ordner ausgewählt. Der Standardwert der Option ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- disable-format-checks

  --disable-format-checks = `true|false`
  Deaktiviert die durch das format-Attribut erzwungene semantische Validierung. Der Standardwert ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- strict-integer-checks

  `--strict-integer-checks =` `true|false`
  Gibt an, ob bei neueren Schemas, in denen die Ganzzahlüberprüfung weniger streng gehandhabt wird, die strengere Ganzzahlüberprüfung aus draft-04 verwendet werden soll. So ist z.B. `1.0` in draft-04 keine gültige Ganzzahl, in späteren Drafts jedoch schon. Diese Option hat keine Auswirkung auf draft-04-Schemas. Der Standardwert der Option ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- Kataloge und globale Ressourcen
  - catalog

    `--catalog =` *FILE*
    Gibt den absoluten Pfad zu einer Root-Katalog-Datei an, die nicht die installierte Root-Katalog-Datei ist. Der Standardwert ist der absolute Pfad zur installierten Root-Katalog-Datei. (`<installationsordner>\Altova\RaptorXMLServer2025\etc\RootCatalog.xml`). Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge 48.

  - user-catalog

    `--user-catalog =` *FILE*
    Definiert den absoluten Pfad zu einem XML-Katalog, der zusätzlich zum Root-Katalog verwendet werden soll. Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge 48.

  - enable-globalresources

    `--enable-globalresources =` `true|false`
    Aktiviert die globalen Ressourcen 55. Standardwert ist `false`.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

  - globalresourceconfig [gc]

    `--gc | --globalresourceconfig =` *VALUE*
    Definiert die aktive Konfiguration der globalen Ressource 55 (und aktiviert globale Ressourcen) 55.

  - globalresourcefile [gr]

    `--gr | --globalresourcefile =` *FILE*
    Definiert die globale Ressourcendatei 55 (und aktiviert globale Ressourcen) 55.

- Allgemeine Optionen
  - error-format

    `--error-format =` `text|shortxml|longxml`
    Definiert das Format der Fehlerausgabe. Der Standardwert ist `text`. Mit den anderen Optionen werden XML-Formate generiert, wobei mit `longxml` mehr Details generiert werden.

- error-limit

  --error-limit = *N | unlimited*
  Definiert das Fehlerlimit mit einem Wertebereich von `1` bis `9999` oder `unbegrenzt`. Der Standardwert ist `100`. Bei Erreichung des Fehlerlimits wird die Validierung gestoppt. Dient dazu, die Prozessorverwendung während der Validierung/Transformation einzuschränken.

- info-limit

  --info-limit = *N | unlimited*
  Definiert die Grenze für Informationsmeldungen im Bereich von `1-65535` oder als `unlimited` (unbegrenzt). Die Verarbeitung wird auch nach Erreichen des Info-Limits fortgesetzt, doch werden weitere Meldungen nicht mehr ausgegeben. Der Standardwert ist `100`.

- help

  --help
  Zeigt den Hilfetext zum Befehl an. Beispiel: `valany --h`. (Alternativ dazu kann der Befehl `help` zusammen mit einem Argument verwendet werden. Beispiel: `help valany`.)

- listfile

  --listfile = `true|false`
  Bei `true` wird das Argument *InputFile* des Befehls als Textdatei behandelt, die einen Dateinamen pro Zeile enthält. Der Standardwert ist `false`. (Als Alternative können die Dateien im CLI getrennt durch ein Leerzeichen aufgelistet werden. Beachten Sie allerdings, dass CLIs eine maximale Zeichenanzahl haben). Beachten Sie, dass die Option `--listfile` nur auf Argumente, nicht aber auf Optionen angewendet wird.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- log-output

  --log-output = *FILE*
  Schreibt die Meldungsausgabe in die angegebene URL. Stellen Sie sicher, dass das CLI Schreibrechte für den Ausgabepfad hat.

- network-timeout

  --network-timeout = *VALUE*
  Definiert das Timeout für entfernte I/O-Operationen in Millisekunden. Der Standardwert ist: `40000`.

- recurse

  --recurse = `true|false`
  Dient zur Auswahl von Dateien innerhalb von Unterverzeichnissen einschließlich ZIP-Archiven. Bei `true` wählt das Argument *InputFile* des Befehls die angegebene Datei auch in den Unterverzeichnissen aus. Beispiel: `"test.zip|zip\test.xml"` wählt Dateien mit dem Namen `test.xml` auf allen Ordnerebenen des ZIP-Ordners aus. Referenzen auf ZIP-Dateien müssen in Anführungszeichen angegeben werden. Es können die Platzhalter `*` und `?` verwendet werden. Mit `*.xml` werden folglich alle `.xml` Dateien im (ZIP-)Ordner ausgewählt. Der Standardwert der Option ist `false`.

Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- verbose

  `--verbose =` `true|false`
  Mit dem Wert `true` wird die Ausgabe zusätzlicher Informationen bei der Validierung aktiviert. Der Standardwert ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- verbose-output

  `--verbose-output =` *`FILE`*
  Schreibt die ausführliche Ausgabe in *`FILE`*.

- version

  `--version`
  Zeigt die Version von RaptorXML Server an. Setzen Sie `--version` bei Verwendung mit einem Befehl vor den Befehl.

- warning-limit

  `--warning-limit =` *`N | unlimited`*
  Definiert das Warnungslimit im Bereich von `1-65535` oder `unbegrenzt`. Bei Erreichen dieses Limits wird die Verarbeitung fortgesetzt, doch werden keine weiteren Warnungen mehr ausgegeben. Der Standardwert ist `100`.

5.5.10 wfjson

Der Befehl **`wfjson`** überprüft die Wohlgeformtheit eines oder mehrerer JSON-Dokumente ECMA-404-Spezifikationen.

```
raptorxml wfjson [options] InputFile
```

- Beim Argument *`InputFile`* handelt es sich um das zu validierende JSON-Dokument (Schemadatei oder Instanzdokument).
- Um mehrere Dokumente zu validieren, (i) listen Sie die zu überprüfenden Dateien entweder in der Befehlszeilenschnittstelle auf, wobei die einzelnen Dateien durch ein Leerzeichen voneinander getrennt sein müssen oder (ii) listen Sie die zu überprüfenden Dateien in einer Textdatei (`.txt`-Datei) auf und zwar einen Dateinamen pro Zeile und geben Sie den Namen dieser Textdatei zusammen mit der auf `true` gesetzten Option `--listfile` 252 als `InputFile`-Argument an (*siehe Liste der Optionen unten*).

Beispiele

Beispiele für den Befehl **wfjson**:

- **raptorxml** wfjson c:\MyJSONFile.json
- **raptorxml** wfjson c:\MyJSONFile-01.json c:\MyJSONFile-02.json
- **raptorxml** wfjson --listfile=true c:\FileList.txt

▼ Groß- und Kleinschreibung und Schrägstriche in der Befehlszeile

RaptorXML (und **RaptorXMLServer** für Administrator-Befehle) *unter Windows*
raptorxml und **raptorxmlserver** für Administrator-Befehle) *unter Windows und Unix (Linux, Mac)*

* Beachten Sie, dass klein geschriebene Befehle (raptorxml und raptorxmlserver) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (RaptorXML) nur unter Windows und Mac ausgeführt werden.
*Verwenden Sie auf Linux und Mac-Systemen Schrägstriche und auf Windows-Systemen umgekehrte Schrägstriche.

▼ Umgekehrte Schrägstriche, Leerzeichen und Sonderzeichen auf Windows-Systemen

Auf Windows-Systemen: Verwenden Sie bei Vorhandensein von Leerzeichen oder Sonerzeichen in Strings (z.B. in Datei- oder Ordnernamen oder Firmen-, Personen- oder Produktnamen) Anführungszeichen: z.B., **"Meine Datei"**. Beachten Sie jedoch, dass ein von einem doppelten Anführungszeichen gefolgter umgekehrter Schrägstrich (z.B: **"C:\Mein Verzeichnis\"**) eventuell nicht korrekt gelesen wird, da der umgekehrte Schrägstrich auch den Beginn einer Escape-Sequenz markiert und die Escape-Sequenz **\"** für ein doppeltes Anführungszeichen steht. Wenn Sie diese Zeichensequenz mit einem Escape versehen wollen, verwenden Sie einen vorangestellten umgekehrten Schrägstrich, wie den folgenden: **\\"**. Einfacher ausgedrückt: Wenn Sie einen Dateipfad, der Leerzeichen oder einen umgekehrten Schrägstrich am Ende enthält, schreiben müssen, so schreiben Sie diesen folgendermaßen: **"C:\Mein Verzeichnis\\"**.

Optionen

Optionen werden in ihrer kurzen Form (falls verfügbar) und in ihrer langen Form aufgelistet. Für die kurze und die lange Form können ein oder zwei Bindestriche verwendet werden. Eine Option kann, muss aber keinen Wert erhalten. Eine Option, die einen Wert erhält, wird folgendermaßen geschrieben: **--option=wert**. Werte können außer in zwei Fällen ohne Anführungszeichen definiert werden: (i) wenn der Wertestring Leerzeichen enthält oder (ii) wenn in der Beschreibung der Option explizit erwähnt ist, dass Anführungszeichen zwingend erforderlich sind. Wenn eine Option einen Booleschen Wert erhält und kein Wert definiert ist, so ist der Standardwert der Option TRUE. Mit Hilfe der Option **--h, --help** können Sie Informationen über den Befehl anzeigen.

▼ Validierung und Verarbeitung

▼ json5

--json5 = true|false
Aktiviert die JSON5-Unterstützung. Der Standardwert ist false.
Hinweis: Die Booleschen Optionswerte werden auf true gesetzt, wenn die Option ohne einen Wert definiert wird.

- jsonc

  `--jsonc =` `true|false`
  Aktiviert die Unterstützung für Kommentare in JSON-Dokumenten. Der Standardwert ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- json-lines

  `--json-lines =` `true|false`
  Aktiviert die Unterstützung für JSON Lines (d.h. einen JSON-Wert pro Zeile). Der Standardwert ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- listfile

  `--listfile =` `true|false`
  Bei `true` wird das Argument *`InputFile`* des Befehls als Textdatei behandelt, die einen Dateinamen pro Zeile enthält. Der Standardwert ist `false`. (Als Alternative können die Dateien im CLI getrennt durch ein Leerzeichen aufgelistet werden. Beachten Sie allerdings, dass CLIs eine maximale Zeichenanzahl haben). Beachten Sie, dass die Option `--listfile` nur auf Argumente, nicht aber auf Optionen angewendet wird.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- recurse

  `--recurse =` `true|false`
  Dient zur Auswahl von Dateien innerhalb von Unterverzeichnissen einschließlich ZIP-Archiven. Bei `true` wählt das Argument *`InputFile`* des Befehls die angegebene Datei auch in den Unterverzeichnissen aus. Beispiel: `"test.zip|zip\test.xml"` wählt Dateien mit dem Namen `test.xml` auf allen Ordnerebenen des ZIP-Ordners aus. Referenzen auf ZIP-Dateien müssen in Anführungszeichen angegeben werden. Es können die Platzhalter `*` und `?` verwendet werden. Mit `*.xml` werden folglich alle `.xml` Dateien im (ZIP-)Ordner ausgewählt. Der Standardwert der Option ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- Kataloge und globale Ressourcen

  - catalog

    `--catalog =` *`FILE`*
    Gibt den absoluten Pfad zu einer Root-Katalog-Datei an, die nicht die installierte Root-Katalog-Datei ist. Der Standardwert ist der absolute Pfad zur installierten Root-Katalog-Datei. (`<installationsordner>\Altova\RaptorXMLServer2025\etc\RootCatalog.xml`). Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge 48.

  - user-catalog

    `--user-catalog =` *`FILE`*
    Definiert den absoluten Pfad zu einem XML-Katalog, der zusätzlich zum Root-Katalog verwendet

werden soll. Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge[48].

- enable-globalresources

  --enable-globalresources = `true|false`
  Aktiviert die globalen Ressourcen[55]. Standardwert ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- globalresourceconfig [gc]

  --gc | --globalresourceconfig = *VALUE*
  Definiert die aktive Konfiguration der globalen Ressource[55] (und aktiviert globale Ressourcen)[55].

- globalresourcefile [gr]

  --gr | --globalresourcefile = *FILE*
  Definiert die globale Ressourcendatei[55] (und aktiviert globale Ressourcen)[55].

- Allgemeine Optionen
  - error-format

    --error-format = `text|shortxml|longxml`
    Definiert das Format der Fehlerausgabe. Der Standardwert ist `text`. Mit den anderen Optionen werden XML-Formate generiert, wobei mit `longxml` mehr Details generiert werden.

  - error-limit

    --error-limit = *N | unlimited*
    Definiert das Fehlerlimit mit einem Wertebereich von `1` bis `9999` oder `unbegrenzt`. Der Standardwert ist `100`. Bei Erreichung des Fehlerlimits wird die Validierung gestoppt. Dient dazu, die Prozessorverwendung während der Validierung/Transformation einzuschränken.

  - info-limit

    --info-limit = *N | unlimited*
    Definiert die Grenze für Informationsmeldungen im Bereich von `1-65535` oder als `unlimited` (unbegrenzt). Die Verarbeitung wird auch nach Erreichen des Info-Limits fortgesetzt, doch werden weitere Meldungen nicht mehr ausgegeben. Der Standardwert ist `100`.

  - help

    --help
    Zeigt den Hilfetext zum Befehl an. Beispiel: `valany --h`. (Alternativ dazu kann der Befehl `help` zusammen mit einem Argument verwendet werden. Beispiel: `help valany`.)

  - listfile

    --listfile = `true|false`
    Bei `true` wird das Argument *InputFile* des Befehls als Textdatei behandelt, die einen Dateinamen pro Zeile enthält. Der Standardwert ist `false`. (Als Alternative können die Dateien im CLI getrennt

durch ein Leerzeichen aufgelistet werden. Beachten Sie allerdings, dass CLIs eine maximale Zeichenanzahl haben). Beachten Sie, dass die Option `--listfile` nur auf Argumente, nicht aber auf Optionen angewendet wird.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- log-output

  `--log-output =` *`FILE`*
  Schreibt die Meldungsausgabe in die angegebene URL. Stellen Sie sicher, dass das CLI Schreibrechte für den Ausgabepfad hat.

- network-timeout

  `--network-timeout =` *`VALUE`*
  Definiert das Timeout für entfernte I/O-Operationen in Millisekunden. Der Standardwert ist: `40000`.

- recurse

  `--recurse =` `true|false`
  Dient zur Auswahl von Dateien innerhalb von Unterverzeichnissen einschließlich ZIP-Archiven. Bei `true` wählt das Argument *`InputFile`* des Befehls die angegebene Datei auch in den Unterverzeichnissen aus. Beispiel: `"test.zip|zip\test.xml"` wählt Dateien mit dem Namen `test.xml` auf allen Ordnerebenen des ZIP-Ordners aus. Referenzen auf ZIP-Dateien müssen in Anführungszeichen angegeben werden. Es können die Platzhalter `*` und `?` verwendet werden. Mit `*.xml` werden folglich alle `.xml` Dateien im (ZIP-)Ordner ausgewählt. Der Standardwert der Option ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- verbose

  `--verbose =` `true|false`
  Mit dem Wert `true` wird die Ausgabe zusätzlicher Informationen bei der Validierung aktiviert. Der Standardwert ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- verbose-output

  `--verbose-output =` *`FILE`*
  Schreibt die ausführliche Ausgabe in *`FILE`*.

- version

  `--version`
  Zeigt die Version von RaptorXML Server an. Setzen Sie `--version` bei Verwendung mit einem Befehl vor den Befehl.

- warning-limit

  `--warning-limit =` *`N | unlimited`*
  Definiert das Warnungslimit im Bereich von `1-65535` oder `unbegrenzt`. Bei Erreichen dieses Limits

wird die Verarbeitung fortgesetzt, doch werden keine weiteren Warnungen mehr ausgegeben. Der Standardwert ist `100`.

5.5.11 wfyaml

Der Befehl **`wfyaml`** überprüft die Wohlgeformtheit eines oder mehrerer YAML-Dokumente gemäß der YAML 1.2-Spezifikation.

```
raptorxml wfyaml [options] InputFile
```

- Das Argument *`InputFile`* ist das YAML-Dokument, das auf Wohlgeformtheit überprüft werden soll.
- Um mehrere Dokumente zu überprüfen, (i) listen Sie entweder die zu überprüfenden Dateien im CLI auf, wobei jede Datei durch ein Leerzeichen von der nächsten getrennt wird, oder (ii) listen Sie die zu überprüfenden Dateien in einer Textdatei (`.txt`-Datei) auf (ein Dateiname pro Zeile) und geben Sie diese Textdatei zusammen mit der auf `true` gesetzten Option `--listfile` 252 als das *`InputFile`* Argument an (*siehe Optionsliste unten*).

Beispiele

Beispiele für den Befehl **`wfyaml`**:

- **`raptorxml`** `wfyaml c:\MyYAMLFile.yaml`
- **`raptorxml`** `wfyaml c:\MyYAMLFile-01.yaml c:\MyYAMLFile-02.yaml`
- **`raptorxml`** `wfyaml --listfile=true c:\FileList.txt`

▼ Groß- und Kleinschreibung und Schrägstriche in der Befehlszeile

`RaptorXML` (und **`RaptorXMLServer`** für Administrator-Befehle) *unter Windows*
`raptorxml` und **`raptorxmlserver`** für Administrator-Befehle) *unter Windows und Unix (Linux, Mac)*

* Beachten Sie, dass klein geschriebene Befehle (`raptorxml` und `raptorxmlserver`) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (`RaptorXML`) nur unter Windows und Mac ausgeführt werden.
*Verwenden Sie auf Linux und Mac-Systemen Schrägstriche und auf Windows-Systemen umgekehrte Schrägstriche.

▼ Umgekehrte Schrägstriche, Leerzeichen und Sonderzeichen auf Windows-Systemen

Auf Windows-Systemen: Verwenden Sie bei Vorhandensein von Leerzeichen oder Sonerzeichen in Strings (z.B. in Datei- oder Ordnernamen oder Firmen-, Personen- oder Produktnamen) Anführungszeichen: z.B., **`"Meine Datei"`**. Beachten Sie jedoch, dass ein von einem doppelten Anführungszeichen gefolgter umgekehrter Schrägstrich (z.B: **`"C:\Mein Verzeichnis\"`**) eventuell nicht korrekt gelesen wird, da der umgekehrte Schrägstrich auch den Beginn einer Escape-Sequenz markiert und die Escape-Sequenz **`\"`** für ein doppeltes Anführungszeichen steht. Wenn Sie diese Zeichensequenz mit einem Escape versehen wollen, verwenden Sie einen vorangestellten umgekehrten Schrägstrich, wie den folgenden: **`\\"`**. Einfacher ausgedrückt: Wenn Sie einen Dateipfad, der Leerzeichen oder einen umgekehrten Schrägstrich am Ende enthält, schreiben müssen, so schreiben Sie diesen

folgendermaßen: `"C:\Mein Verzeichnis\\"`.

Optionen

Optionen werden in ihrer kurzen Form (falls verfügbar) und in ihrer langen Form aufgelistet. Für die kurze und die lange Form können ein oder zwei Bindestriche verwendet werden. Eine Option kann, muss aber keinen Wert erhalten. Eine Option, die einen Wert erhält, wird folgendermaßen geschrieben: **`--option=wert`**. Werte können außer in zwei Fällen ohne Anführungszeichen definiert werden: (i) wenn der Wertestring Leerzeichen enthält oder (ii) wenn in der Beschreibung der Option explizit erwähnt ist, dass Anführungszeichen zwingend erforderlich sind. Wenn eine Option einen Booleschen Wert erhält und kein Wert definiert ist, so ist der Standardwert der Option `TRUE`. Mit Hilfe der Option **`--h, --help`** können Sie Informationen über den Befehl anzeigen.

- Kataloge und globale Ressourcen
  - catalog

    `--catalog =` *FILE*
    Gibt den absoluten Pfad zu einer Root-Katalog-Datei an, die nicht die installierte Root-Katalog-Datei ist. Der Standardwert ist der absolute Pfad zur installierten Root-Katalog-Datei. (`<installationsordner>\Altova\RaptorXMLServer2025\etc\RootCatalog.xml`). Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge[48].

  - user-catalog

    `--user-catalog =` *FILE*
    Definiert den absoluten Pfad zu einem XML-Katalog, der zusätzlich zum Root-Katalog verwendet werden soll. Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge[48].

  - enable-globalresources

    `--enable-globalresources =` `true|false`
    Aktiviert die globalen Ressourcen[55]. Standardwert ist `false`.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

  - globalresourceconfig [gc]

    `--gc | --globalresourceconfig =` *VALUE*
    Definiert die aktive Konfiguration der globalen Ressource[55] (und aktiviert globale Ressourcen)[55].

  - globalresourcefile [gr]

    `--gr | --globalresourcefile =` *FILE*
    Definiert die globale Ressourcendatei[55] (und aktiviert globale Ressourcen)[55].

- Allgemeine Optionen
  - error-format

    `--error-format =` `text|shortxml|longxml`
    Definiert das Format der Fehlerausgabe. Der Standardwert ist `text`. Mit den anderen Optionen

werden XML-Formate generiert, wobei mit `longxml` mehr Details generiert werden.

- error-limit

  `--error-limit =` *`N | unlimited`*
  Definiert das Fehlerlimit mit einem Wertebereich von `1` bis `9999` oder `unbegrenzt`. Der Standardwert ist `100`. Bei Erreichung des Fehlerlimits wird die Validierung gestoppt. Dient dazu, die Prozessorverwendung während der Validierung/Transformation einzuschränken.

- info-limit

  `--info-limit =` *`N | unlimited`*
  Definiert die Grenze für Informationsmeldungen im Bereich von `1-65535` oder als `unlimited` (unbegrenzt). Die Verarbeitung wird auch nach Erreichen des Info-Limits fortgesetzt, doch werden weitere Meldungen nicht mehr ausgegeben. Der Standardwert ist `100`.

- help

  `--help`
  Zeigt den Hilfetext zum Befehl an. Beispiel: `valany --h`. (Alternativ dazu kann der Befehl `help` zusammen mit einem Argument verwendet werden. Beispiel: `help valany`.)

- listfile

  `--listfile =` `true|false`
  Bei `true` wird das Argument *`InputFile`* des Befehls als Textdatei behandelt, die einen Dateinamen pro Zeile enthält. Der Standardwert ist `false`. (Als Alternative können die Dateien im CLI getrennt durch ein Leerzeichen aufgelistet werden. Beachten Sie allerdings, dass CLIs eine maximale Zeichenanzahl haben). Beachten Sie, dass die Option `--listfile` nur auf Argumente, nicht aber auf Optionen angewendet wird.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- log-output

  `--log-output =` *`FILE`*
  Schreibt die Meldungsausgabe in die angegebene URL. Stellen Sie sicher, dass das CLI Schreibrechte für den Ausgabepfad hat.

- network-timeout

  `--network-timeout =` *`VALUE`*
  Definiert das Timeout für entfernte I/O-Operationen in Millisekunden. Der Standardwert ist: `40000`.

- recurse

  `--recurse =` `true|false`
  Dient zur Auswahl von Dateien innerhalb von Unterverzeichnissen einschließlich ZIP-Archiven. Bei `true` wählt das Argument *`InputFile`* des Befehls die angegebene Datei auch in den Unterverzeichnissen aus. Beispiel: `"test.zip|zip\test.xml"` wählt Dateien mit dem Namen `test.xml` auf allen Ordnerebenen des ZIP-Ordners aus. Referenzen auf ZIP-Dateien müssen in Anführungszeichen angegeben werden. Es können die Platzhalter `*` und `?` verwendet werden. Mit

`*.xml` werden folglich alle `.xml` Dateien im (ZIP-)Ordner ausgewählt. Der Standardwert der Option ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- verbose

  `--verbose =` `true|false`
  Mit dem Wert `true` wird die Ausgabe zusätzlicher Informationen bei der Validierung aktiviert. Der Standardwert ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- verbose-output

  `--verbose-output =` *`FILE`*
  Schreibt die ausführliche Ausgabe in *`FILE`*.

- version

  `--version`
  Zeigt die Version von RaptorXML Server an. Setzen Sie `--version` bei Verwendung mit einem Befehl vor den Befehl.

- warning-limit

  `--warning-limit =` *`N | unlimited`*
  Definiert das Warnungslimit im Bereich von `1-65535` oder `unbegrenzt`. Bei Erreichen dieses Limits wird die Verarbeitung fortgesetzt, doch werden keine weiteren Warnungen mehr ausgegeben. Der Standardwert ist `100`.

5.5.12 xml2json

Der Befehl **`xml2json`** konvertiert ein XML-Instanzdokument in ein JSON-Dokument.

```
raptorxml XML2json [options] XMLFile
```

- Das Argument *`XMLFile`* ist die zu konvertierende XML-Datei.
- Mit Hilfe der Option `--conversion-output` definieren Sie den Pfad zur generierten JSON-Datei.

Beispiel

Beispiel für den Befehl **`xml2json`**:

- **`raptorxml`** `xml2json --conversion-output=c:\MyJSONData.json c:\MyXMLData.xml`

- Groß- und Kleinschreibung und Schrägstriche in der Befehlszeile

RaptorXML (und **RaptorXMLServer** für Administrator-Befehle) *unter Windows*
raptorxml und **raptorxmlserver** für Administrator-Befehle) *unter Windows und Unix (Linux, Mac)*

* Beachten Sie, dass klein geschriebene Befehle (raptorxml und raptorxmlserver) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (RaptorXML) nur unter Windows und Mac ausgeführt werden.
*Verwenden Sie auf Linux und Mac-Systemen Schrägstriche und auf Windows-Systemen umgekehrte Schrägstriche.

▼ Umgekehrte Schrägstriche, Leerzeichen und Sonderzeichen auf Windows-Systemen

Auf Windows-Systemen: Verwenden Sie bei Vorhandensein von Leerzeichen oder Sonerzeichen in Strings (z.B. in Datei- oder Ordnernamen oder Firmen-, Personen- oder Produktnamen) Anführungszeichen: z.B., **"Meine Datei"**. Beachten Sie jedoch, dass ein von einem doppelten Anführungszeichen gefolgter umgekehrter Schrägstrich (z.B: **"C:\Mein Verzeichnis\"**) eventuell nicht korrekt gelesen wird, da der umgekehrte Schrägstrich auch den Beginn einer Escape-Sequenz markiert und die Escape-Sequenz **\"** für ein doppeltes Anführungszeichen steht. Wenn Sie diese Zeichensequenz mit einem Escape versehen wollen, verwenden Sie einen vorangestellten umgekehrten Schrägstrich, wie den folgenden: **\\"**. Einfacher ausgedrückt: Wenn Sie einen Dateipfad, der Leerzeichen oder einen umgekehrten Schrägstrich am Ende enthält, schreiben müssen, so schreiben Sie diesen folgendermaßen: **"C:\Mein Verzeichnis\\"**.

Optionen

Optionen werden in ihrer kurzen Form (falls verfügbar) und in ihrer langen Form aufgelistet. Für die kurze und die lange Form können ein oder zwei Bindestriche verwendet werden. Eine Option kann, muss aber keinen Wert erhalten. Eine Option, die einen Wert erhält, wird folgendermaßen geschrieben: **--option=wert**. Werte können außer in zwei Fällen ohne Anführungszeichen definiert werden: (i) wenn der Wertestring Leerzeichen enthält oder (ii) wenn in der Beschreibung der Option explizit erwähnt ist, dass Anführungszeichen zwingend erforderlich sind. Wenn eine Option einen Booleschen Wert erhält und kein Wert definiert ist, so ist der Standardwert der Option TRUE. Mit Hilfe der Option **--h, --help** können Sie Informationen über den Befehl anzeigen.

▼ Optionen für die Konvertierung von XML in JSON

Mit diesen Optionen wird bei Konvertierungen zwischen XML und JSON die Behandlung bestimmter konvertierungsbezogener Details definiert.

▼ attributes

--attributes = true|false
Bei **true** erfolgt eine Konvertierung zwischen XML-Attributen und JSON-Eigenschaften mit dem Präfix @. Andernfalls werden XML-Attribute und JSON-@-Eigenschaften nicht konvertiert. Der Standardwert ist **true**.
Hinweis: Die Booleschen Optionswerte werden auf true gesetzt, wenn die Option ohne einen Wert definiert wird.

▼ comments

--comments = true|false
Bei **true** erfolgt eine Konvertierung zwischen XML-Kommentaren und JSON-Eigenschaften mit dem Präfix #. Andernfalls werden XML Attribute und JSON-#-Eigenschaften nicht konvertiert. Der Standardwert ist **true**.

Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- conversion-output, o

  `--o, --conversion-output =` `FILE`
  Definiert den Pfad und Namen der Datei, an die das Ergebnis der Konvertierung gesendet wird.

- ignore-pis

  `--ignore-pis =` `true|false`
  Bei **`true`** werden Processing Instructions im XML-Quelldokument ignoriert. Der Standardwert ist **`false`**.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- merge-elements

  `--merge-elements =` `true|false`
  Bei **`true`** wird anhand von Elementen desselben Namens und derselben Ebene des XML-Dokuments im generierten JSON-Dokument ein Array erstellt. Der Standardwert ist **`true`**.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- merge-text

  `--merge-text =` `true|false`
  Bei **`true`** wird anhand von Text-Nodes derselben Ebene des XML-Dokuments im generierten JSON-Dokument ein Array erstellt. Der Standardwert ist **`true`**.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- pi

  `--pi =` `true|false`
  Bei **`true`** erfolgt eine Konvertierung zwischen XML Processing Instructions und JSON-Eigenschaften mit dem Präfix ?. Andernfalls werden XML-Attribute und JSON-?-Eigenschaften nicht konvertiert. Der Standardwert ist **`true`**.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- pretty-print

  `--pp, --pretty-print =` `true|false`
  Bei **`true`** wird das generierte Ausgabedokument mit Pretty-Print formatiert. Der Standardwert ist **`false`**.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- text

  `--text =` `true|false`
  Bei **`true`** erfolgt eine Konvertierung zwischen XML-Textinhalt und JSON-Eigenschaften mit dem Präfix

$. Andernfalls werden XML-Attribute und JSON-$-Eigenschaften nicht konvertiert. Der Standardwert ist **true**.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

Allgemeine Optionen

error-format

--error-format = `text|shortxml|longxml`
Definiert das Format der Fehlerausgabe. Der Standardwert ist `text`. Mit den anderen Optionen werden XML-Formate generiert, wobei mit `longxml` mehr Details generiert werden.

error-limit

--error-limit = *N | unlimited*
Definiert das Fehlerlimit mit einem Wertebereich von `1` bis `9999` oder `unbegrenzt`. Der Standardwert ist `100`. Bei Erreichung des Fehlerlimits wird die Validierung gestoppt. Dient dazu, die Prozessorverwendung während der Validierung/Transformation einzuschränken.

info-limit

--info-limit = *N | unlimited*
Definiert die Grenze für Informationsmeldungen im Bereich von `1-65535` oder als `unlimited` (unbegrenzt). Die Verarbeitung wird auch nach Erreichen des Info-Limits fortgesetzt, doch werden weitere Meldungen nicht mehr ausgegeben. Der Standardwert ist `100`.

help

--help
Zeigt den Hilfetext zum Befehl an. Beispiel: `valany --h`. (Alternativ dazu kann der Befehl `help` zusammen mit einem Argument verwendet werden. Beispiel: `help valany`.)

listfile

--listfile = `true|false`
Bei `true` wird das Argument *InputFile* des Befehls als Textdatei behandelt, die einen Dateinamen pro Zeile enthält. Der Standardwert ist `false`. (Als Alternative können die Dateien im CLI getrennt durch ein Leerzeichen aufgelistet werden. Beachten Sie allerdings, dass CLIs eine maximale Zeichenanzahl haben). Beachten Sie, dass die Option `--listfile` nur auf Argumente, nicht aber auf Optionen angewendet wird.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

log-output

--log-output = *FILE*
Schreibt die Meldungsausgabe in die angegebene URL. Stellen Sie sicher, dass das CLI Schreibrechte für den Ausgabepfad hat.

network-timeout

--network-timeout = *VALUE*
Definiert das Timeout für entfernte I/O-Operationen in Millisekunden. Der Standardwert ist: `40000`.

- recurse

  --recurse = `true|false`
  Dient zur Auswahl von Dateien innerhalb von Unterverzeichnissen einschließlich ZIP-Archiven. Bei `true` wählt das Argument *InputFile* des Befehls die angegebene Datei auch in den Unterverzeichnissen aus. Beispiel: `"test.zip|zip\test.xml"` wählt Dateien mit dem Namen `test.xml` auf allen Ordnerebenen des ZIP-Ordners aus. Referenzen auf ZIP-Dateien müssen in Anführungszeichen angegeben werden. Es können die Platzhalter `*` und `?` verwendet werden. Mit `*.xml` werden folglich alle `.xml` Dateien im (ZIP-)Ordner ausgewählt. Der Standardwert der Option ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- verbose

  --verbose = `true|false`
  Mit dem Wert `true` wird die Ausgabe zusätzlicher Informationen bei der Validierung aktiviert. Der Standardwert ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- verbose-output

  --verbose-output = *FILE*
  Schreibt die ausführliche Ausgabe in *FILE*.

- version

  --version
  Zeigt die Version von RaptorXML Server an. Setzen Sie `--version` bei Verwendung mit einem Befehl vor den Befehl.

- warning-limit

  --warning-limit = *N | unlimited*
  Definiert das Warnungslimit im Bereich von `1-65535` oder `unbegrenzt`. Bei Erreichen dieses Limits wird die Verarbeitung fortgesetzt, doch werden keine weiteren Warnungen mehr ausgegeben. Der Standardwert ist `100`.

- Kataloge und globale Ressourcen
  - catalog

    --catalog = *FILE*
    Gibt den absoluten Pfad zu einer Root-Katalog-Datei an, die nicht die installierte Root-Katalog-Datei ist. Der Standardwert ist der absolute Pfad zur installierten Root-Katalog-Datei. (`<installationsordner>\Altova\RaptorXMLServer2025\etc\RootCatalog.xml`). Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge[48].

- user-catalog

  --user-catalog = *FILE*
  Definiert den absoluten Pfad zu einem XML-Katalog, der zusätzlich zum Root-Katalog verwendet werden soll. Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge[48].

- enable-globalresources

  --enable-globalresources = `true|false`
  Aktiviert die globalen Ressourcen[55]. Standardwert ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- globalresourceconfig [gc]

  --gc | --globalresourceconfig = *VALUE*
  Definiert die aktive Konfiguration der globalen Ressource[55] (und aktiviert globale Ressourcen)[55].

- globalresourcefile [gr]

  --gr | --globalresourcefile = *FILE*
  Definiert die globale Ressourcendatei[55] (und aktiviert globale Ressourcen)[55].

5.5.13 xsd2jsonschema

Der Befehl **xsd2jsonschema** konvertiert ein oder mehrere W3C XML Schema 1.0- oder 1.1-Dokumente in ein JSON-Schema-Dokument.

```
raptorxml xsd2jsonschema [options] XSDFile
```

- Das Argument *XSDFile* ist die zu konvertierende XML-Schema-Datei.
- Mit Hilfe der Option `--schema-conversion-output` definieren Sie den Pfad zur generierten XSD-Datei.

Beispiel

Beispiel für den Befehl **xsd2jsonschema**:

- **raptorxml** `xsd2jsonschema --schema-conversion-output=c:\MyJSONSchema.json c:\MyXSDSchema.xsd`

- Groß- und Kleinschreibung und Schrägstriche in der Befehlszeile

  RaptorXML (und **RaptorXMLServer** für Administrator-Befehle) *unter Windows*
  raptorxml und **raptorxmlserver** für Administrator-Befehle) *unter Windows und Unix (Linux, Mac)*

  * Beachten Sie, dass klein geschriebene Befehle (`raptorxml` und `raptorxmlserver`) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (`RaptorXML`) nur unter

Windows und Mac ausgeführt werden.
*Verwenden Sie auf Linux und Mac-Systemen Schrägstriche und auf Windows-Systemen umgekehrte Schrägstriche.

▼ Umgekehrte Schrägstriche, Leerzeichen und Sonderzeichen auf Windows-Systemen

Auf Windows-Systemen: Verwenden Sie bei Vorhandensein von Leerzeichen oder Sonerzeichen in Strings (z.B. in Datei- oder Ordnernamen oder Firmen-, Personen- oder Produktnamen) Anführungszeichen: z.B., **`"Meine Datei"`**. Beachten Sie jedoch, dass ein von einem doppelten Anführungszeichen gefolgter umgekehrter Schrägstrich (z.B: **`"C:\Mein Verzeichnis\"`**) eventuell nicht korrekt gelesen wird, da der umgekehrte Schrägstrich auch den Beginn einer Escape-Sequenz markiert und die Escape-Sequenz **`\"`** für ein doppeltes Anführungszeichen steht. Wenn Sie diese Zeichensequenz mit einem Escape versehen wollen, verwenden Sie einen vorangestellten umgekehrten Schrägstrich, wie den folgenden: **`\\"`**. Einfacher ausgedrückt: Wenn Sie einen Dateipfad, der Leerzeichen oder einen umgekehrten Schrägstrich am Ende enthält, schreiben müssen, so schreiben Sie diesen folgendermaßen: **`"C:\Mein Verzeichnis\\"`**.

Optionen

Optionen werden in ihrer kurzen Form (falls verfügbar) und in ihrer langen Form aufgelistet. Für die kurze und die lange Form können ein oder zwei Bindestriche verwendet werden. Eine Option kann, muss aber keinen Wert erhalten. Eine Option, die einen Wert erhält, wird folgendermaßen geschrieben: **`--option=wert`**. Werte können außer in zwei Fällen ohne Anführungszeichen definiert werden: (i) wenn der Wertestring Leerzeichen enthält oder (ii) wenn in der Beschreibung der Option explizit erwähnt ist, dass Anführungszeichen zwingend erforderlich sind. Wenn eine Option einen Booleschen Wert erhält und kein Wert definiert ist, so ist der Standardwert der Option `TRUE`. Mit Hilfe der Option **`--h, --help`** können Sie Informationen über den Befehl anzeigen.

▼ XML-Schema-Definitionsoptionen

▼ schema-imports

`--schema-imports =` `load-by-schemalocation | load-preferring-schemalocation | load-by-namespace | load-combining-both | license-namespace-only`
Definiert das Verhalten von `xs:import` Elementen, von denen jedes ein optionales `namespace` Attribut und ein optionales `schemaLocation` Attribut hat: `<import namespace="someNS" schemaLocation="someURL">`. Mit der Option wird definiert, ob ein Schema-Dokument geladen oder nur ein Namespace lizenziert werden soll und, wenn ein Schema-Dokument geladen werden soll, anhand welcher Informationen es gesucht werden soll. Standardeinstellung: `load-preferring-schemalocation`. Das Verhalten ist das folgende:

- `load-by-schemalocation`: Anhand des Werts des `schemaLocation` Attributs wird der Schemapfad ermittelt, wobei Katalog-Mappings[48] berücksichtigt werden. Wenn das Namespace-Attribut vorhanden ist, wird der Namespace importiert (lizenziert).
- `load-preferring-schemalocation`: Wenn das `schemaLocation` Attribut vorhanden ist, wird es verwendet, wobei Katalog-Mappings[48] berücksichtigt werden. Falls kein `schemaLocation` Attribut vorhanden ist, wird der Wert des `namespace` Attributs über ein Katalog-Mapping[48] verwendet. Dies ist der **Standardwert**.
- `load-by-namespace`: Anhand des Werts des `namespace` Attributs wird der Schemapfad über ein Katalog-Mapping[48] ermittelt.
- `load-combining-both`: Wenn entweder das Attribut `namespace` oder das Attribut `schemaLocation` ein Katalog-Mapping[48] hat, so wird das Mapping verwendet. Wenn beide

Attribute Katalog-Mappings[48] haben, ist es vom Wert der Option `--schema-mapping` (XML/XSD-Option[255]) abhängig, welches Mapping verwendet wird. Falls kein Katalog-Mapping[48] vorhanden ist, wird das `schemaLocation` Attribut verwendet.

- `license-namespace-only`: Der Namespace wird importiert. Kein Schema-Dokument wird importiert.

▼ schema-mapping

`--schema-mapping =` `prefer-schemalocation | prefer-namespace`

Wenn ein Schema-Dokument sowohl anhand des Schema-Pfads als auch des Namespace gesucht werden soll, wird damit festgelegt, welche der beiden Optionen beim Katalog-Lookup Vorrang erhält. (Wenn eine der Optionen `--schemalocation-hints` oder `--schema-imports` einen Wert `load-combining-both` hat und wenn die betroffenen Namespace- und URL-Teile beide Katalog-Mappings[48] haben, gibt der Wert dieser Option an, welches der beiden Mappings verwendet werden soll (das Namespace Mapping oder das URL-Mapping; der Wert `prefer-schemalocation` bezieht sich auf das URL-Mapping)). Der Standardwert ist `prefer-schemalocation`.

▼ schema-location-hints

`--schema-location-hints =` `load-by-schemalocation | load-by-namespace | load-combining-both | ignore`

Definiert das Verhalten der Attribute `xsi:schemaLocation` und `xsi:noNamespaceSchemaLocation`: Legt fest, ob ein Schema-Dokument geladen werden soll und falls ja, anhand welcher Informationen es gesucht werden soll. Standardeinstellung: `load-by-schemalocation`.

- Der Wert `load-by-schemalocation` verwendet die URL des Schemapfads[422] in den Attributen `xsi:schemaLocation` und `xsi:noNamespaceSchemaLocation` in XML-Instanzdokumenten. Dies ist der **Standardwert**.
- Der Wert `load-by-namespace` verwendet den Namespace-Teil[422] von `xsi:schemaLocation` und - im Fall von `xsi:noNamespaceSchemaLocation` - einen leeren String und ermittelt das Schema über ein Katalog-Mapping[48].
- Bei Verwendung von `load-combining-both` und wenn entweder der Namespace-Teil oder der URL-Teil ein Katalog-Mapping[48] hat, so wird das Katalog-Mapping[48] verwendet. Wenn beide Katalog-Mappings[48] haben, ist es vom Wert der `--schema-mapping` Option (XML/XSD-Option[255]) abhängig, welches Mapping verwendet wird. Wenn weder der Namespace noch die URL ein Katalog-Mapping hat, wird die URL verwendet.
- Wenn der Wert der Option `ignore` ist, werden die beiden Attribute `xsi:schemaLocation` und `xsi:noNamespaceSchemaLocation` ignoriert.

▼ XML-Schema-Verarbeitungsoptionen

▼ report-import-namespace-mismatch-as-warning

`--report-import-namespace-mismatch-as-warning =` `true|false`

Stuft Fehler, die beim Import von Schemas mit `xs:import` aufgrund eines nicht übereinstimmenden Namespace oder Ziel-Namespace auftreten, von Fehlern auf Warnungen herab. Der Standardwert ist `false`.

Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

▼ xinclude

`--xinclude =` `true|false`

Aktiviert die Unterstützung für XML-Inkludierungen (XInclude). Der Standardwert ist `false`. Bei `false` werden die `include`-Elemente von XInclude ignoriert.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- xml-mode-for-schemas

  `--xml-mode-for-schemas =` `wf|id|valid`
  Definiert, welcher XML-Verarbeitungsmodus für XML-Schema-Dokumente verwendet werden soll: `wf`=Wohlgeformtheitsprüfung; `id`=Wohlgeformtheitsprüfung mit ID/IDREF-Prüfung; `valid`=Validierung. Der Standardwert ist `wf`. Beachten Sie, dass jedes bei der Verarbeitung geladene Schema-Dokument eine DTD referenzieren muss, damit der Wert `valid` ausgegeben werden kann. Falls keine DTD vorhanden ist, wird ein Fehler ausgegeben.

- xsd-version

  `--xsd-version =` `1.0|1.1|detect`
  Definiert die zu verwendende W3C Schema Definition Language (XSD) Version. Der Standardwert ist `1.0`. Diese Option eignet sich auch, um herauszufinden, inwiefern ein 1.0-kompatibles Schema nicht mit Schemaversion 1.1 kompatibel ist. Die Option `detect` ist eine Altova-spezifische Funktionalität. Mit dieser Option kann die Version des XML-Schema-Dokuments (`1.0` oder `1.1`) durch Lesen des Werts des `vc:minVersion` Attributs des `<xs:schema>` Elements des Dokuments ermittelt werden. Wenn der Wert des `@vc:minVersion` Attributs `1.1` ist, wird das Schema als Version `1.1` erkannt. Bei jedem anderen Wert wird das Schema als `1.0` erkannt. Bei jedem anderen Wert oder bei Fehlen des `@vc:minVersion` Attributs wird das Schema als Version `1.0` gelesen.

- Konvertierung von XSD in JSON-Schema
  - array-and-item

    `--array-and-item =` `true|false`
    Bei **`true`** sind im generierten JSON-Schema nicht nur Arrays, sondern auch einzelne Elemente für Partikel mit `maxOccurs > 1` zulässig. Der Standardwert ist **`true`**.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

  - consider-format

    `--consider-format =` `true|false`
    Wenn die Option auf **`true`** gesetzt ist, werden die Datentypen des Quellschemas, wenn möglich, in die entsprechenden Typen des Zielschemas konvertiert. Der Standardwert ist **`false`**.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

  - jsonschema-version

    `--jsonschema-version =` `draft04|draft06|draft07|2019-09|2020-12|oas-3.1|latest|detect`
    Definiert, welche Version der JSON-Schema-Draft-Spezifikation verwendet werden soll. Der Standardwert ist `detect`.

  - property-for-comments

`--property-for-comments =` `true|false`
Bei **`true`** wird in den einzelnen Subschemas eine Eigenschaft namens `'#'` erstellt, damit Kommentare unterstützt werden. Der Standardwert ist **`false`**.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- property-for-pis

`--property-for-pis =` `true|false`
Bei **`true`** wird eine Mustereigenschaft erstellt, die Eigenschaften mit dem Präfix `'?'` in den einzelnen Subschemas entspricht, damit XML Processing Instructions unterstützt werden. Der Standardwert ist **`false`**.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- property-for-xmlns

`--property-for-xmlns =` `true|false`
Bei **`true`** wird eine Mustereigenschaft erstellt, die Eigenschaften mit dem Präfix `'@xmlns'` in den einzelnen Subschemas entspricht, damit Namespace-Deklarationen unterstützt werden. Der Standardwert ist **`false`**.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- property-for-xsi

`--property-for-xsi =` `true|false`
Bei **`true`** wird eine Mustereigenschaft erstellt, die Eigenschaften mit dem Präfix `'@xsi'` in den einzelnen Subschemas entspricht, damit `xsi:*`-Attribute wie z.B. `xsi:schemaLocation` unterstützt werden. Der Standardwert ist **`false`**.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- simple-content-pure-object

`--simple-content-pure-object=` `true|false`
Bei **`true`** wird ein "pure" Objekt mit einfachem Inhalt für komplexe Typen erstellt. Der Standardwert ist **`false`**.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- schema-conversion-output, o

`--o, --schema-converson-output =` `FILE`
Definiert den Pfad und Namen der Datei, an die das Ergebnis der Konvertierung gesendet wird.

- simplify-occurrence-constraints

`--simplify-occurrence-constraints =` `true|false`
Bei **`true`** werden (i) Occurrence-Definitionen im XML-Schema im JSON-Schema zu entweder "obligatorisch" oder "optional" (ii) und sich wiederholende Elemente im XML-Schema werden zu Arrays mit unbegrenzten **`maxItems`** vereinfacht. Der Standardwert ist **`false`**.

Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- xmlns

  `--xml-mode =` `wf|id|valid`
  Definiert URI-Präfix-Mappings für die Namespaces im XML-Schema.

- Allgemeine Optionen
  - error-format

    `--error-format =` `text|shortxml|longxml`
    Definiert das Format der Fehlerausgabe. Der Standardwert ist `text`. Mit den anderen Optionen werden XML-Formate generiert, wobei mit `longxml` mehr Details generiert werden.

  - error-limit

    `--error-limit =` *`N | unlimited`*
    Definiert das Fehlerlimit mit einem Wertebereich von `1` bis `9999` oder `unbegrenzt`. Der Standardwert ist `100`. Bei Erreichung des Fehlerlimits wird die Validierung gestoppt. Dient dazu, die Prozessorverwendung während der Validierung/Transformation einzuschränken.

  - info-limit

    `--info-limit =` *`N | unlimited`*
    Definiert die Grenze für Informationsmeldungen im Bereich von `1-65535` oder als `unlimited` (unbegrenzt). Die Verarbeitung wird auch nach Erreichen des Info-Limits fortgesetzt, doch werden weitere Meldungen nicht mehr ausgegeben. Der Standardwert ist `100`.

  - help

    `--help`
    Zeigt den Hilfetext zum Befehl an. Beispiel: `valany --h`. (Alternativ dazu kann der Befehl `help` zusammen mit einem Argument verwendet werden. Beispiel: `help valany`.)

  - listfile

    `--listfile =` `true|false`
    Bei `true` wird das Argument *`InputFile`* des Befehls als Textdatei behandelt, die einen Dateinamen pro Zeile enthält. Der Standardwert ist `false`. (Als Alternative können die Dateien im CLI getrennt durch ein Leerzeichen aufgelistet werden. Beachten Sie allerdings, dass CLIs eine maximale Zeichenanzahl haben). Beachten Sie, dass die Option `--listfile` nur auf Argumente, nicht aber auf Optionen angewendet wird.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

  - log-output

    `--log-output =` *`FILE`*
    Schreibt die Meldungsausgabe in die angegebene URL. Stellen Sie sicher, dass das CLI Schreibrechte für den Ausgabepfad hat.

- network-timeout

  --network-timeout = *VALUE*
  Definiert das Timeout für entfernte I/O-Operationen in Millisekunden. Der Standardwert ist: `40000`.

- recurse

  --recurse = `true|false`
  Dient zur Auswahl von Dateien innerhalb von Unterverzeichnissen einschließlich ZIP-Archiven. Bei `true` wählt das Argument *InputFile* des Befehls die angegebene Datei auch in den Unterverzeichnissen aus. Beispiel: `"test.zip|zip\test.xml"` wählt Dateien mit dem Namen `test.xml` auf allen Ordnerebenen des ZIP-Ordners aus. Referenzen auf ZIP-Dateien müssen in Anführungszeichen angegeben werden. Es können die Platzhalter `*` und `?` verwendet werden. Mit `*.xml` werden folglich alle `.xml` Dateien im (ZIP-)Ordner ausgewählt. Der Standardwert der Option ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- verbose

  --verbose = `true|false`
  Mit dem Wert `true` wird die Ausgabe zusätzlicher Informationen bei der Validierung aktiviert. Der Standardwert ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- verbose-output

  --verbose-output = *FILE*
  Schreibt die ausführliche Ausgabe in *FILE*.

- version

  --version
  Zeigt die Version von RaptorXML Server an. Setzen Sie `--version` bei Verwendung mit einem Befehl vor den Befehl.

- warning-limit

  --warning-limit = *N | unlimited*
  Definiert das Warnungslimit im Bereich von `1-65535` oder `unbegrenzt`. Bei Erreichen dieses Limits wird die Verarbeitung fortgesetzt, doch werden keine weiteren Warnungen mehr ausgegeben. Der Standardwert ist `100`.

- Kataloge und globale Ressourcen

  - catalog

    --catalog = *FILE*
    Gibt den absoluten Pfad zu einer Root-Katalog-Datei an, die nicht die installierte Root-Katalog-Datei ist. Der Standardwert ist der absolute Pfad zur installierten Root-Katalog-Datei. (`<installationsordner>\Altova\RaptorXMLServer2025\etc\RootCatalog.xml`). Informationen

zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge[48].

- user-catalog

  --user-catalog = *FILE*
  Definiert den absoluten Pfad zu einem XML-Katalog, der zusätzlich zum Root-Katalog verwendet werden soll. Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge[48].

- enable-globalresources

  --enable-globalresources = `true|false`
  Aktiviert die globalen Ressourcen[55]. Standardwert ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- globalresourceconfig [gc]

  --gc | --globalresourceconfig = *VALUE*
  Definiert die aktive Konfiguration der globalen Ressource[55] (und aktiviert globale Ressourcen)[55].

- globalresourcefile [gr]

  --gr | --globalresourcefile = *FILE*
  Definiert die globale Ressourcendatei[55] (und aktiviert globale Ressourcen)[55].

5.6 XML-Signaturbefehle

Mit Hilfe der XML-Signaturbefehle kann ein XML-Dokument signiert und ein signiertes Dokument überprüft werden. Unten sehen Sie eine Liste dieser Befehle, die in den Unterabschnitten dieses Abschnitts näher beschrieben sind.

- xmlsignature-sign 208: Erstellt anhand eines Input-Dokuments ein XML-Signatur-Ausgabedokument.
- xmlsignature-verify 213: Überprüft ein XML-Signaturdokument.
- xmlsignature-update 208: Aktualisiert die Signatur eines (geänderten) XML-Dokuments.
- xmlsignature-remove 208: Entfernt die Signatur eines XML-Dokuments.

5.6.1 xmlsignature-sign

Der Befehl **`xmlsignature-sign | xsign`** erhält ein XML-Dokument als Input und erstellt unter Verwendung der definierten Signaturoptionen ein XML-Signatur-Ausgabedokument.

```
EXENAME-LC%> xmlsignature-sign [options] --output=File --signature-type=Value --
signature-canonicalization-method=Value --certname=Value|hmackey=Value InputFile
```

- Das Argument *`InputFile`* steht für das zu signierende XML-Dokument.
- Mit der Option **`--output`** wird der Pfad des Dokuments, das die XML-Signatur enthält, definiert.

Beispiele

Beispiele für den Befehl **`xmlsignature-sign`**:

- **`raptorxml`** `xsign --output=c:\SignedFile.xml --signature-type=enveloped --signature-canonicalization-method=xml-c14n11 --hmackey=secretpassword c:\SomeUnsigned.xml`

▼ Groß- und Kleinschreibung und Schrägstriche in der Befehlszeile

`RaptorXML` (und **`RaptorXMLServer`** für Administrator-Befehle) *unter Windows*
`raptorxml` und **`raptorxmlserver`** für Administrator-Befehle) *unter Windows und Unix (Linux, Mac)*

* Beachten Sie, dass klein geschriebene Befehle (`raptorxml` und `raptorxmlserver`) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (`RaptorXML`) nur unter Windows und Mac ausgeführt werden.
*Verwenden Sie auf Linux und Mac-Systemen Schrägstriche und auf Windows-Systemen umgekehrte Schrägstriche.

▼ Umgekehrte Schrägstriche, Leerzeichen und Sonderzeichen auf Windows-Systemen

Auf Windows-Systemen: Verwenden Sie bei Vorhandensein von Leerzeichen oder Sonerzeichen in Strings (z.B. in Datei- oder Ordnernamen oder Firmen-, Personen- oder Produktnamen) Anführungszeichen: z.B., **`"Meine Datei"`**. Beachten Sie jedoch, dass ein von einem doppelten Anführungszeichen gefolgter umgekehrter Schrägstrich (z.B: **`"C:\Mein Verzeichnis\"`**) eventuell nicht korrekt gelesen wird, da der umgekehrte Schrägstrich auch den Beginn einer Escape-Sequenz markiert und die Escape-Sequenz **`\"`** für ein doppeltes Anführungszeichen steht. Wenn Sie diese Zeichensequenz mit einem Escape versehen wollen, verwenden Sie einen vorangestellten umgekehrten Schrägstrich, wie

den folgenden: `\\"`. Einfacher ausgedrückt: Wenn Sie einen Dateipfad, der Leerzeichen oder einen umgekehrten Schrägstrich am Ende enthält, schreiben müssen, so schreiben Sie diesen folgendermaßen: `"C:\Mein Verzeichnis\\"`.

Optionen

Optionen werden in ihrer kurzen Form (falls verfügbar) und in ihrer langen Form aufgelistet. Für die kurze und die lange Form können ein oder zwei Bindestriche verwendet werden. Eine Option kann, muss aber keinen Wert erhalten. Eine Option, die einen Wert erhält, wird folgendermaßen geschrieben: **`--option=wert`**. Werte können außer in zwei Fällen ohne Anführungszeichen definiert werden: (i) wenn der Wertestring Leerzeichen enthält oder (ii) wenn in der Beschreibung der Option explizit erwähnt ist, dass Anführungszeichen zwingend erforderlich sind. Wenn eine Option einen Booleschen Wert erhält und kein Wert definiert ist, so ist der Standardwert der Option `TRUE`. Mit Hilfe der Option **`--h, --help`** können Sie Informationen über den Befehl anzeigen.

- Allgemeine Optionen
  - output

    `output =` *FILE*
    Die URL des Ausgabedokuments, das mit der neuen XML-Signatur erstellt wird.

  - verbose

    `--verbose =` `true|false`
    Mit dem Wert `true` wird die Ausgabe zusätzlicher Informationen bei der Validierung aktiviert. Der Standardwert ist `false`.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- XML-Signaturoptionen
  - absolute-reference-uri

    `--absolute-reference-uri =` `true|false`
    Definiert, ob die URI des signierten Dokuments als absolute (`true`) oder relative (`false`) URI gelesen werden soll. Der Standardwert ist `false`.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

  - certname, certificate-name

    `--certname, --certificate-name =` *VALUE*
    Der Name des zum Signieren verwendeten Zertifikats.

    Windows
    Dies ist der **Subject-**Name eines Zertifikats aus dem ausgewählten **`--certificate-store`** (Zertifikatspeicher).

    eispiel zum Aufliste der Zertifikate (unter PowerShell)

    ```
    % ls cert://CurrentUser/My
    ```

```
PSParentPath: Microsoft.PowerShell.Security\Certificate::CurrentUser\My
Thumbprint Subject
---------- -------
C9DF64BB0AAF5FA73474D78B7CCFFC37C95BFC6C CN=certificate1
... CN=...
```

eispiel: **--certificate-name==certificate1**

Linux/MacOS
--certname definiert den Dateinamen eines PEM-kodierten X.509v3-Zertifikats mit dem Private Key. Solche Dateien haben die Erweiterung **.pem**.

eispiel: **--certificate-name==/path/to/certificate1.pem**

- certstore, certificate-store

  --certstore, --certificate-store = *VALUE*
  Der Pfad, unter dem das mit **--certificate-name** definierte Zertifikat gespeichert ist.

  Windows
  Der Name eines Zertifikatspeichers unter **cert://CurrentUser**. Die verfügbaren Zertifikatspeicher können mit Hilfe von **% ls cert://CurrentUser/** (unter PowerShell) aufgelistet werden. Die Zertifikate würden anschließend folgendermaßen aufgelistet:

```
Name : TrustedPublisher
Name : ClientAuthIssuer
Name : Root
Name : UserDS
Name : CA
Name : ACRS
Name : REQUEST
Name : AuthRoot
Name : MSIEHistoryJournal
Name : TrustedPeople
Name : MyCertStore
Name : Local NonRemovable Certificates
Name : SmartCardRoot
Name : Trust
Name : Disallowed
```

  eispiel: **--certificate-store==MyCertStore**

  Linux/MacOS
  Die Option **--certstore** wird derzeit nicht unterstützt.

- digest, digest-method

  --digest, --digest-method = sha1|sha256|sha384|sha512|base64
  Der zur Berechnung des Digest-Werts an der XML-Input-Datei angewendete Algorithmus. Verfügbare

Werte sind: `sha1|sha256|sha384|sha512`.

- hmackey, hmac-secret-key

  `--hmackey, --hmac-secret-key =` *VALUE*
  Der Shared Secret HMAC-Schlüssel; muss mindestens sechs Zeichen lang sein.

  Beispiel: **`--hmackey=secretpassword`**

- hmaclen, hmac-output-length

  `--hmaclen, --hmac-output-length =` *LENGTH*
  Kürzt die Ausgabe des HMAC-Algorithmus auf `length` Bits. Falls definiert, muss dieser Wert
  - ein Vielfaches von 8 sein
  - größer als 80 sein
  - mehr als die Hälfte der Ausgabelänge des zugrunde liegenden Hash-Algorithmus betragen

- keyinfo, append-keyinfo

  `--keyinfo, --append-keyinfo =` `true|false`
  Definiert, ob das **`KeyInfo`**-Element in der Signatur inkludiert werden soll oder nicht. Der Standardwert ist **`false`**.

- sigc14nmeth, signature-canonicalization-method

  `--sigc14nmeth, --signature-canonicalization-method =` *VALUE*
  Definiert, welcher Kanonisierungsalgorithmus auf das Element **`SignedInfo`** angewendet werden soll. Der Wert muss einer der folgenden sein:

  - `REC-xml-c14n-20010315`
  - `xml-c14n11`
  - `xml-exc-c14n#`

- sigmeth, signature-method

  `--sigmeth, --signature-method =` *VALUE*
  Definiert, welcher Algorithmus zum Generieren der Signatur verwendet werden soll.

  Wenn ein Zertifikat verwendet wird
  Wenn ein Zertifikat definiert ist, dann ist `SignatureMethod` optional und der Wert dieses Parameters wird vom Zertifikat abgeleitet. Wenn die Option definiert ist, muss sie mit dem vom Zertifikat verwendeten Algorithmus übereinstimmen.
  eispiel: `rsa-sha256`.

  Wenn --hmac-secret-key verwendet wird
  Wenn `HMACSecretKey` verwendet wird, ist diese `SignatureMethod` obligatorisch. Der Wert muss einer der unterstützten HMAC-Algorithmen sein:
  - `hmac-sha256`
  - `hmac-sha386`
  - `hmac-sha512`
  - hmac-sha1 (soll laut Spezifikation nicht verwendet werden)

eispiel: **hmac-sha256**

- sigtype, signature-type

  --sigtype, --signature-type = `detached | enveloping | enveloped`
  Definiert den Typ der zu generierenden Signatur.

- transforms

  --transforms = *VALUE*
  Definiert, welche XML-Signaturtransformation auf das Dokument angewendet werden soll. Die unterstützten Werte sind die folgenden:

  - `REC-xml-c14n-20010315` für Canonical XML 1.0 (Kommentare weglassen)
  - `xml-c14n11` für Canonical XML 1.1 (Kommentare weglassen)
  - `xml-exc-c14n#` für Exclusive XML Canonicalization 1.0 (Kommentare weglassen)
  - `REC-xml-c14n-20010315#WithComments` für Canonical XML 1.0 (mit Kommentaren)
  - `xml-c14n11#WithComments` for Canonical XML 1.1 (mit Kommentaren)
  - `xml-exc-c14n#WithComments` for Exclusive XML Canonicalization 1.0 (mit Kommentaren)
  - `base64`
  - `strip-whitespaces` Altova-Erweiterung

  eispiel: **--transforms=xml-c14n11**

  Anmerkung: Diese Option kann mehrmals definiert werden. Falls sie mehrmals definiert ist, spielt die Reihenfolge, in der sie definiert ist, eine Rolle. Die erste definierte Transformation gilt für das Input-Dokument. Die letzte definierte Transformation wird unmittelbar vor der Berechnung des Digest-Werts verwendet.

- write-default-attributes

  --write-default-attributes = `true|false`
  Definiert, ob Standardattributwerte aus der DTD im signierten Dokument inkludiert werden sollen.

- Optionen Hilfe und Version
  - help

    --help
    Zeigt den Hilfetext zum Befehl an. Beispiel: `valany --h`. (Alternativ dazu kann der Befehl `help` zusammen mit einem Argument verwendet werden. Beispiel: `help valany`.)

  - version

    --version
    Zeigt die Version von RaptorXML Server an. Setzen Sie `--version` bei Verwendung mit einem Befehl vor den Befehl.

5.6.2 xmlsignature-verify

Mit dem Befehl **xmlsignature-verify | xverify** wird die XML-Signatur der Input-Datei überprüft.

```
raptorxml xmlsignature-verify [options] InputFile
```

- Das Argument *InputFile* steht für das zu überprüfende signierte XML-Dokument.
- Wenn die Überprüfung erfolgreich ist, wird die Meldung **result="OK"** angezeigt, andernfalls wird die Meldung **result="Failed"** angezeigt.

Beispiel

Beispiele für den Befehl **xmlsignature-verify**:

- **raptorxml** xverify c:\SignedFile.xml

▼ Groß- und Kleinschreibung und Schrägstriche in der Befehlszeile

RaptorXML (und **RaptorXMLServer** für Administrator-Befehle) *unter Windows*
raptorxml und **raptorxmlserver** für Administrator-Befehle) *unter Windows und Unix (Linux, Mac)*

* Beachten Sie, dass klein geschriebene Befehle (raptorxml und raptorxmlserver) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (RaptorXML) nur unter Windows und Mac ausgeführt werden.
*Verwenden Sie auf Linux und Mac-Systemen Schrägstriche und auf Windows-Systemen umgekehrte Schrägstriche.

▼ Umgekehrte Schrägstriche, Leerzeichen und Sonderzeichen auf Windows-Systemen

Auf Windows-Systemen: Verwenden Sie bei Vorhandensein von Leerzeichen oder Sonerzeichen in Strings (z.B. in Datei- oder Ordnernamen oder Firmen-, Personen- oder Produktnamen) Anführungszeichen: z.B., **"Meine Datei"**. Beachten Sie jedoch, dass ein von einem doppelten Anführungszeichen gefolgter umgekehrter Schrägstrich (z.B: **"C:\Mein Verzeichnis\"**) eventuell nicht korrekt gelesen wird, da der umgekehrte Schrägstrich auch den Beginn einer Escape-Sequenz markiert und die Escape-Sequenz **\"** für ein doppeltes Anführungszeichen steht. Wenn Sie diese Zeichensequenz mit einem Escape versehen wollen, verwenden Sie einen vorangestellten umgekehrten Schrägstrich, wie den folgenden: **\\"**. Einfacher ausgedrückt: Wenn Sie einen Dateipfad, der Leerzeichen oder einen umgekehrten Schrägstrich am Ende enthält, schreiben müssen, so schreiben Sie diesen folgendermaßen: **"C:\Mein Verzeichnis\\"**.

Optionen

Optionen werden in ihrer kurzen Form (falls verfügbar) und in ihrer langen Form aufgelistet. Für die kurze und die lange Form können ein oder zwei Bindestriche verwendet werden. Eine Option kann, muss aber keinen Wert erhalten. Eine Option, die einen Wert erhält, wird folgendermaßen geschrieben: **--option=wert**. Werte können außer in zwei Fällen ohne Anführungszeichen definiert werden: (i) wenn der Wertestring Leerzeichen enthält oder (ii) wenn in der Beschreibung der Option explizit erwähnt ist, dass Anführungszeichen zwingend erforderlich sind. Wenn eine Option einen Booleschen Wert erhält und kein Wert definiert ist, so ist der Standardwert der Option TRUE. Mit Hilfe der Option **--h, --help** können Sie Informationen über den Befehl anzeigen.

- Allgemeine Optionen
  - verbose

    --verbose = `true|false`
    Mit dem Wert `true` wird die Ausgabe zusätzlicher Informationen bei der Validierung aktiviert. Der Standardwert ist `false`.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- XML-Signatur-Optionen
  - certname, certificate-name

    --certname, --certificate-name = *VALUE*
    Der Name des zum Signieren verwendeten Zertifikats.

    Windows
    Dies ist der **Subject-**Name eines Zertifikats aus dem ausgewählten **--certificate-store** (Zertifikatspeicher).

    eispiel zum Aufliste der Zertifikate (unter PowerShell)
    ```
    % ls cert://CurrentUser/My
    PSParentPath: Microsoft.PowerShell.Security\Certificate::CurrentUser\My
    Thumbprint Subject
    ---------- -------
    C9DF64BB0AAF5FA73474D78B7CCFFC37C95BFC6C CN=certificate1
    ... CN=...
    ```

    eispiel: **--certificate-name==certificate1**

    Linux/MacOS
    --certname definiert den Dateinamen eines PEM-kodierten X.509v3-Zertifikats mit dem Private Key. Solche Dateien haben die Erweiterung **.pem**.

    eispiel: **--certificate-name==/path/to/certificate1.pem**

  - certstore, certificate-store

    --certstore, --certificate-store = *VALUE*
    Der Pfad, unter dem das mit **--certificate-name** definierte Zertifikat gespeichert ist.

    Windows
    Der Name eines Zertifikatspeichers unter **cert://CurrentUser**. Die verfügbaren Zertifikatspeicher können mit Hilfe von **% ls cert://CurrentUser/** (unter PowerShell) aufgelistet werden. Die Zertifikate würden anschließend folgendermaßen aufgelistet:

    ```
    Name : TrustedPublisher
    Name : ClientAuthIssuer
    ```



```
Name : Root
Name : UserDS
Name : CA
Name : ACRS
Name : REQUEST
Name : AuthRoot
Name : MSIEHistoryJournal
Name : TrustedPeople
Name : MyCertStore
Name : Local NonRemovable Certificates
Name : SmartCardRoot
Name : Trust
Name : Disallowed
```

eispiel: **--certificate-store==MyCertStore**

Linux/MacOS
Die Option **--certstore** wird derzeit nicht unterstützt.

- hmackey, hmac-secret-key

  --hmackey, --hmac-secret-key = *VALUE*
  Der Shared Secret HMAC-Schlüssel; muss mindestens sechs Zeichen lang sein.

  Beispiel: **--hmackey=secretpassword**

- ignore-certificate-errors

  --i, --ignore-certificate-errors = `true|false`
  Wenn die Option auf **true** gesetzt ist, werden Zertifikatfehler bei der Überprüfung von XML-Signaturen (die **SignedInfo**-Elemente) in einem XML-Dokument ignoriert. Der Standardwert ist **false**.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- Optionen Hilfe und Version
  - help

    --help
    Zeigt den Hilfetext zum Befehl an. Beispiel: `valany --h`. (Alternativ dazu kann der Befehl `help` zusammen mit einem Argument verwendet werden. Beispiel: `help valany`.)

  - version

    --version
    Zeigt die Version von RaptorXML Server an. Setzen Sie `--version` bei Verwendung mit einem Befehl vor den Befehl.

5.6.3 xmlsignature-update

Mit dem Befehl **xmlsignature-update | xupdate** wird die XML-Signatur in der signierten Input-Datei aktualisiert. Wenn das Dokument geändert wurde, ist die aktualisierte XML-Signatur eine andere. Andernfalls ist die aktualisierte Signatur mit der vorherigen Signatur identisch.

```
raptorxml xmlsignature-update [options] --output=File SignedFile
```

- Das Argument *SignedFile* steht für das signierte zu aktualisierende XML-Dokument.
- Es müssen entweder die Option (i) `hmac-secret-key` oder (ii) die Optionen `certificate-name` und `certificate-store` definiert werden.
- Wenn die Optionen `certificate-name` und `certificate-store` definiert sind, müssen diese mit denjenigen übereinstimmen, die zuvor zum Signieren des XML-Dokuments verwendet wurden. (Beachten Sie, dass die Option `certificate-store` derzeit unter Linux und macOS nicht unterstützt wird.)

Beispiele

Beispiele für den Befehl **xmlsignature-update**:

- **raptorxml** xupdate --output=c:\UpdatedSignedFile.xml --certname=certificate1 --certstore=MyCertStore c:\SomeSignedFile.xml
- **raptorxml** xupdate --output=c:\UpdatedSignedFile.xml --hmackey=SecretPassword c:\SomeSignedFile.xml

▼ Groß- und Kleinschreibung und Schrägstriche in der Befehlszeile

RaptorXML (und **RaptorXMLServer** für Administrator-Befehle) *unter Windows*
raptorxml und **raptorxmlserver** für Administrator-Befehle) *unter Windows und Unix (Linux, Mac)*

* Beachten Sie, dass klein geschriebene Befehle (`raptorxml` und `raptorxmlserver`) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (`RaptorXML`) nur unter Windows und Mac ausgeführt werden.
*Verwenden Sie auf Linux und Mac-Systemen Schrägstriche und auf Windows-Systemen umgekehrte Schrägstriche.

▼ Umgekehrte Schrägstriche, Leerzeichen und Sonderzeichen auf Windows-Systemen

Auf Windows-Systemen: Verwenden Sie bei Vorhandensein von Leerzeichen oder Sonerzeichen in Strings (z.B. in Datei- oder Ordnernamen oder Firmen-, Personen- oder Produktnamen) Anführungszeichen: z.B., **"Meine Datei"**. Beachten Sie jedoch, dass ein von einem doppelten Anführungszeichen gefolgter umgekehrter Schrägstrich (z.B: **"C:\Mein Verzeichnis\"**) eventuell nicht korrekt gelesen wird, da der umgekehrte Schrägstrich auch den Beginn einer Escape-Sequenz markiert und die Escape-Sequenz **\"** für ein doppeltes Anführungszeichen steht. Wenn Sie diese Zeichensequenz mit einem Escape versehen wollen, verwenden Sie einen vorangestellten umgekehrten Schrägstrich, wie den folgenden: **\\"**. Einfacher ausgedrückt: Wenn Sie einen Dateipfad, der Leerzeichen oder einen umgekehrten Schrägstrich am Ende enthält, schreiben müssen, so schreiben Sie diesen folgendermaßen: **"C:\Mein Verzeichnis\\"**.

Optionen

Optionen werden in ihrer kurzen Form (falls verfügbar) und in ihrer langen Form aufgelistet. Für die kurze und die lange Form können ein oder zwei Bindestriche verwendet werden. Eine Option kann, muss aber keinen Wert erhalten. Eine Option, die einen Wert erhält, wird folgendermaßen geschrieben: **`--option=wert`**. Werte können außer in zwei Fällen ohne Anführungszeichen definiert werden: (i) wenn der Wertestring Leerzeichen enthält oder (ii) wenn in der Beschreibung der Option explizit erwähnt ist, dass Anführungszeichen zwingend erforderlich sind. Wenn eine Option einen Booleschen Wert erhält und kein Wert definiert ist, so ist der Standardwert der Option `TRUE`. Mit Hilfe der Option **`--h, --help`** können Sie Informationen über den Befehl anzeigen.

- Allgemeine Optionen
  - output

    `output =` *FILE*
    Die URL des Ausgabedokuments, das mit der neuen XML-Signatur erstellt wird.

  - verbose

    `--verbose =` `true|false`
    Mit dem Wert `true` wird die Ausgabe zusätzlicher Informationen bei der Validierung aktiviert. Der Standardwert ist `false`.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- XML-Signatur-Optionen
  - certname, certificate-name

    `--certname, --certificate-name =` *VALUE*
    Der Name des zum Signieren verwendeten Zertifikats.

    Windows
    Dies ist der **Subject-**Name eines Zertifikats aus dem ausgewählten **`--certificate-store`** (Zertifikatspeicher).

    eispiel zum Aufliste der Zertifikate (unter PowerShell)

    ```
    % ls cert://CurrentUser/My
    PSParentPath: Microsoft.PowerShell.Security\Certificate::CurrentUser\My
    Thumbprint Subject
    ---------- -------
    C9DF64BB0AAF5FA73474D78B7CCFFC37C95BFC6C CN=certificate1
    ... CN=...
    ```

    eispiel: **`--certificate-name==certificate1`**

    Linux/MacOS
    `--certname` definiert den Dateinamen eines PEM-kodierten X.509v3-Zertifikats mit dem Private Key. Solche Dateien haben die Erweiterung **`.pem`**.

eispiel: **--certificate-name==/path/to/certificate1.pem**

- certstore, certificate-store

  --certstore, --certificate-store = *VALUE*
  Der Pfad, unter dem das mit **--certificate-name** definierte Zertifikat gespeichert ist.

  Windows
  Der Name eines Zertifikatspeichers unter **cert://CurrentUser**. Die verfügbaren Zertifikatspeicher können mit Hilfe von **% ls cert://CurrentUser/** (unter PowerShell) aufgelistet werden. Die Zertifikate würden anschließend folgendermaßen aufgelistet:

```
Name : TrustedPublisher
Name : ClientAuthIssuer
Name : Root
Name : UserDS
Name : CA
Name : ACRS
Name : REQUEST
Name : AuthRoot
Name : MSIEHistoryJournal
Name : TrustedPeople
Name : MyCertStore
Name : Local NonRemovable Certificates
Name : SmartCardRoot
Name : Trust
Name : Disallowed
```

  eispiel: **--certificate-store==MyCertStore**

  Linux/MacOS
  Die Option **--certstore** wird derzeit nicht unterstützt.

- hmackey, hmac-secret-key

  --hmackey, --hmac-secret-key = *VALUE*
  Der Shared Secret HMAC-Schlüssel; muss mindestens sechs Zeichen lang sein.

  Beispiel: **--hmackey=secretpassword**

- Optionen Hilfe und Version
  - help

    --help
    Zeigt den Hilfetext zum Befehl an. Beispiel: `valany --h`. (Alternativ dazu kann der Befehl `help` zusammen mit einem Argument verwendet werden. Beispiel: `help valany`.)

  - version

--version
Zeigt die Version von RaptorXML Server an. Setzen Sie `--version` bei Verwendung mit einem Befehl vor den Befehl.

5.6.4 xmlsignature-remove

Der Befehl **xmlsignature-remove | xremove** entfernt die XML-Signatur der signierten Input-Datei und speichert das nicht signierte Dokument unter einem von Ihnen gespeicherten Ausgabepfad.

```
raptorxml xmlsignature-remove [options] --output=File SignedFile
```

- Das Argument *SignedFile* ist das signierte XML-Dokument, aus dem die XML-Signatur entfernt werden soll.
- Die Option `--output` definiert den Pfad des nicht signierten XML-Dokuments, das generiert wird.

Beispiele

Beispiele für den Befehl **xmlsignature-remove**:

- **raptorxml** xremove --output=c:\UnsignedFile.xml c:\SignedFile.xml

▼ Groß- und Kleinschreibung und Schrägstriche in der Befehlszeile

RaptorXML (und **RaptorXMLServer** für Administrator-Befehle) *unter Windows*
raptorxml und **raptorxmlserver** für Administrator-Befehle) *unter Windows und Unix (Linux, Mac)*

* Beachten Sie, dass klein geschriebene Befehle (`raptorxml` und `raptorxmlserver`) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (`RaptorXML`) nur unter Windows und Mac ausgeführt werden.
*Verwenden Sie auf Linux und Mac-Systemen Schrägstriche und auf Windows-Systemen umgekehrte Schrägstriche.

▼ Umgekehrte Schrägstriche, Leerzeichen und Sonderzeichen auf Windows-Systemen

Auf Windows-Systemen: Verwenden Sie bei Vorhandensein von Leerzeichen oder Sonerzeichen in Strings (z.B. in Datei- oder Ordnernamen oder Firmen-, Personen- oder Produktnamen) Anführungszeichen: z.B., **"Meine Datei"**. Beachten Sie jedoch, dass ein von einem doppelten Anführungszeichen gefolgter umgekehrter Schrägstrich (z.B: **"C:\Mein Verzeichnis\"**) eventuell nicht korrekt gelesen wird, da der umgekehrte Schrägstrich auch den Beginn einer Escape-Sequenz markiert und die Escape-Sequenz **\"** für ein doppeltes Anführungszeichen steht. Wenn Sie diese Zeichensequenz mit einem Escape versehen wollen, verwenden Sie einen vorangestellten umgekehrten Schrägstrich, wie den folgenden: **\\"**. Einfacher ausgedrückt: Wenn Sie einen Dateipfad, der Leerzeichen oder einen umgekehrten Schrägstrich am Ende enthält, schreiben müssen, so schreiben Sie diesen folgendermaßen: **"C:\Mein Verzeichnis\\"**.

Optionen

Optionen werden in ihrer kurzen Form (falls verfügbar) und in ihrer langen Form aufgelistet. Für die kurze und die lange Form können ein oder zwei Bindestriche verwendet werden. Eine Option kann, muss aber keinen Wert erhalten. Eine Option, die einen Wert erhält, wird folgendermaßen geschrieben: **`--option=wert`**. Werte können außer in zwei Fällen ohne Anführungszeichen definiert werden: (i) wenn der Wertestring Leerzeichen enthält oder (ii) wenn in der Beschreibung der Option explizit erwähnt ist, dass Anführungszeichen zwingend erforderlich sind. Wenn eine Option einen Booleschen Wert erhält und kein Wert definiert ist, so ist der Standardwert der Option `TRUE`. Mit Hilfe der Option **`--h, --help`** können Sie Informationen über den Befehl anzeigen.

- Allgemeine Optionen
  - output

    `output =` *FILE*
    Die URL des Ausgabedokuments, das unter Entfernung der XML-Signatur generiert wird.

  - verbose

    `--verbose =` `true|false`
    Mit dem Wert `true` wird die Ausgabe zusätzlicher Informationen bei der Validierung aktiviert. Der Standardwert ist `false`.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- Optionen Hilfe und Version
  - help

    `--help`
    Zeigt den Hilfetext zum Befehl an. Beispiel: `valany --h`. (Alternativ dazu kann der Befehl `help` zusammen mit einem Argument verwendet werden. Beispiel: `help valany`.)

  - version

    `--version`
    Zeigt die Version von RaptorXML Server an. Setzen Sie `--version` bei Verwendung mit einem Befehl vor den Befehl.

5.7 Allgemeine Befehle

Dieser Abschnitt enthält eine Beschreibung der folgenden allgemeinen Befehle:

- valany[221]: Validiert das angegebene Dokument anhand seines Typs.
- script[222]: Führt ein Python Skript aus.
- help[223]: Zeigt Informationen zum genannten Befehl an.

5.7.1 valany

Der Befehl **valany** ist ein allgemeiner Befehl, der ein Dokument anhand des jeweiligen Dokumenttyps validiert. Der Typ des Dokuments wird automatisch erkannt und die jeweilige Validierung erfolgt anhand der entsprechenden Spezifikation. Das Argument *InputFile* ist das zu validierende Dokument. Beachten Sie, dass nur ein Dokument als das Argument des Befehls angegeben werden kann.

```
raptorxml valany [options] InputFile
```

Der Befehl **valany** kann für die folgenden Validierungsarten verwendet werden. Seine Optionen sind diejenigen, die für den jeweiligen Einzelvalidierungsbefehl zur Verfügung stehen. Eine Liste der jeweiligen Optionen finden Sie in der Beschreibung zum jeweiligen Validierungsbefehl.

- valdtd (dtd)[72]
- valxsd (xsd)[77]
- valxml-withdtd (xml)[60]
- valxml-withxsd (xsi)[65]
- valxslt[139]
- valxquery[115]
- valavrojson (avrojson)[164]

Beispiele

- **raptorxml** valany c:\Test.xsd

▼ Groß- und Kleinschreibung und Schrägstriche in der Befehlszeile

RaptorXML (und **RaptorXMLServer** für Administrator-Befehle) *unter Windows*
raptorxml und **raptorxmlserver** für Administrator-Befehle) *unter Windows und Unix (Linux, Mac)*

* Beachten Sie, dass klein geschriebene Befehle (raptorxml und raptorxmlserver) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (RaptorXML) nur unter Windows und Mac ausgeführt werden.
*Verwenden Sie auf Linux und Mac-Systemen Schrägstriche und auf Windows-Systemen umgekehrte Schrägstriche.

▼ Umgekehrte Schrägstriche, Leerzeichen und Sonderzeichen auf Windows-Systemen

Auf Windows-Systemen: Verwenden Sie bei Vorhandensein von Leerzeichen oder Sonerzeichen in Strings (z.B. in Datei- oder Ordnernamen oder Firmen-, Personen- oder Produktnamen)

Anführungszeichen: z.B., **`"Meine Datei"`**. Beachten Sie jedoch, dass ein von einem doppelten Anführungszeichen gefolgter umgekehrter Schrägstrich (z.B: **`"C:\Mein Verzeichnis\"`**) eventuell nicht korrekt gelesen wird, da der umgekehrte Schrägstrich auch den Beginn einer Escape-Sequenz markiert und die Escape-Sequenz **`\"`** für ein doppeltes Anführungszeichen steht. Wenn Sie diese Zeichensequenz mit einem Escape versehen wollen, verwenden Sie einen vorangestellten umgekehrten Schrägstrich, wie den folgenden: **`\\"`**. Einfacher ausgedrückt: Wenn Sie einen Dateipfad, der Leerzeichen oder einen umgekehrten Schrägstrich am Ende enthält, schreiben müssen, so schreiben Sie diesen folgendermaßen: **`"C:\Mein Verzeichnis\\"`**.

Optionen

Eine Liste der jeweiligen Optionen finden Sie unter der Beschreibung des jeweiligen Validierungsbefehls. Beachten Sie jedoch, dass der Befehl **`valany`** im Gegensatz zu den meisten Einzelvalidierungsbefehlen, die mehrere Input-Dokumente haben können, nur ein Input-Dokument akzeptiert. Optionen wie die Option `--listfile` sind daher bei **`valany`** nicht anwendbar.

5.7.2 script

Mit dem Befehl **`script`** wird ein Python 3.11.8-Skript ausgeführt, das die RaptorXML Python API verwendet.

```
raptorxml script [options] PythonScriptFile
```

Das Argument *`File`* ist der Pfad zum Python-Skript, das ausgeführt werden soll. Es stehen zusätzliche Optionen zur Verfügung. Eine Liste dieser Optionen erhalten Sie, wenn Sie den folgenden Befehl ausführen:

```
raptorxml script [-h | --help]
```

Beispiele

- **`raptorxml`** `script c:\MyPythonScript.py`
- **`raptorxml`** `script -h`
- **`raptorxml`** `script` `#` *Ohne eine Skript-Datei wird eine interaktive Python Shell gestartet*
- **`raptorxml`** `script -m pip` `#` *Lädt das Modul* `pip` *und führt es aus; siehe Abschnitt "Optionen" weiter unten*

▼ Groß- und Kleinschreibung und Schrägstriche in der Befehlszeile

`RaptorXML` (und **`RaptorXMLServer`** für Administrator-Befehle) *unter Windows*
`raptorxml` und **`raptorxmlserver`** für Administrator-Befehle) *unter Windows und Unix (Linux, Mac)*

* Beachten Sie, dass klein geschriebene Befehle (`raptorxml` und `raptorxmlserver`) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (`RaptorXML`) nur unter Windows und Mac ausgeführt werden.
*Verwenden Sie auf Linux und Mac-Systemen Schrägstriche und auf Windows-Systemen umgekehrte Schrägstriche.

- Umgekehrte Schrägstriche, Leerzeichen und Sonderzeichen auf Windows-Systemen

  Auf Windows-Systemen: Verwenden Sie bei Vorhandensein von Leerzeichen oder Sonerzeichen in Strings (z.B. in Datei- oder Ordnernamen oder Firmen-, Personen- oder Produktnamen) Anführungszeichen: z.B., **`"Meine Datei"`**. Beachten Sie jedoch, dass ein von einem doppelten Anführungszeichen gefolgter umgekehrter Schrägstrich (z.B: **`"C:\Mein Verzeichnis\"`**) eventuell nicht korrekt gelesen wird, da der umgekehrte Schrägstrich auch den Beginn einer Escape-Sequenz markiert und die Escape-Sequenz **`\"`** für ein doppeltes Anführungszeichen steht. Wenn Sie diese Zeichensequenz mit einem Escape versehen wollen, verwenden Sie einen vorangestellten umgekehrten Schrägstrich, wie den folgenden: **`\\"`**. Einfacher ausgedrückt: Wenn Sie einen Dateipfad, der Leerzeichen oder einen umgekehrten Schrägstrich am Ende enthält, schreiben müssen, so schreiben Sie diesen folgendermaßen: **`"C:\Mein Verzeichnis\\"`**.

Optionen

Alle hinter dem Befehl `script` aufgelisteten Optionen und Argumente werden direkt an den Python Interpreter weitergeleitet. Eine vollständige Liste der verfügbaren Optionen finden Sie auf der Python-Dokumentationsseite https://docs.python.org/3.11/using/cmdline.html.

5.7.3 help

Syntax und Beschreibung

Der Befehl **`help`** hat ein einziges Argument (`Command`): den Namen des Befehls, zu dem die Hilfe benötigt wird. Er zeigt die korrekte Syntax des Befehls, seine Optionen sowie andere relevante Informationen an. Wenn das Argument `Command` nicht angegeben wird, werden alle Befehle der ausführbaren Datei aufgelistet, wobei zu jedem eine kurze Textbeschreibung angezeigt wird. Der Befehl **`help`** kann von beiden ausführbaren Dateien aus aufgerufen werden: **`raptorxml`** und **`raptorxmlserver`**.

```
raptorxml help Command
raptorxmlserver help Command
```

- Groß- und Kleinschreibung und Schrägstriche in der Befehlszeile

  `RaptorXML` (und **`RaptorXMLServer`** für Administrator-Befehle) *unter Windows*
  `raptorxml` und **`raptorxmlserver`** für Administrator-Befehle) *unter Windows und Unix (Linux, Mac)*

  * Beachten Sie, dass klein geschriebene Befehle (`raptorxml` und `raptorxmlserver`) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (`RaptorXML`) nur unter Windows und Mac ausgeführt werden.
  *Verwenden Sie auf Linux und Mac-Systemen Schrägstriche und auf Windows-Systemen umgekehrte Schrägstriche.

Beispiel

Beispiele für den Befehl **`help`**, um Informationen über den Befehl **`licenserver`** (Dieser Befehl steht in beiden ausführbaren Dateien zur Verfügung) anzuzeigen:

```
raptorxml help licenseserver
raptorxmlserver help licenseserver
```

Die Option --help

Die Hilfe zu einem Befehl kann auch über die Option `--help` im Anschluss an diesen Befehl aufgerufen werden. Mit den beiden unten stehenden Befehlen erhalten Sie dasselbe Ergebnis:

```
raptorxml licenseserver --help
```

Im obigen Befehl wird die Option `--help` des Befehls **`licenseserver`** verwendet.

```
raptorxml help licenseserver
```

Der Befehl **`help`** erhält `licenseserver` als Argument.

In beiden Fällen wird die Hilfe zum Befehl **`licenseserver`** angezeigt.

5.8 Lokalisierungsbefehle

Sie können für jede Sprache Ihrer Wahl eine lokalisierte Version der RaptorXML Applikation erstellen. Im Ordner `<ProgramFilesFolder>\Altova\RaptorXMLServer2025\bin\` stehen fünf lokalisierte Versionen (Englisch, Deutsch, Spanisch, Französisch und Japanisch) bereits zur Verfügung. Eine Lokalisierung für diese Sprachen ist daher nicht mehr notwendig.

Um eine lokalisierte Version in einer anderen Sprache zu erstellen, gehen Sie folgendermaßen vor:

1. Generieren Sie mit Hilfe des Befehls `exportresourcestrings` [225] eine XML-Datei, die die Ressourcenstrings enthält. Die Ressourcenstrings in dieser XML-Datei sind in einer der fünf unterstützten Sprachen: je nachdem, welches Argument mit dem Befehl verwendet wird, in Englisch (`en`), Deutsch (`de`), Spanisch (`es`), Französisch (`fr`) oder Japanisch (`ja`).
2. Übersetzen Sie die Ressourcenstrings aus der Sprache der generierten XML-Datei in die Zielsprache. Die Ressourcenstrings bilden den Inhalt der `<string>` Elemente in der XML-Datei. Übersetzen Sie keine Variablen in geschweiften Klammern wie z.B. `{option}` oder `{product}`.
3. Wenden Sie sich an den Altova Support, um anhand Ihrer übersetzten XML-Datei eine lokalisierte RaptorXML DLL-Datei zu generieren.
4. Nachdem Sie Ihre lokalisierte DLL-Datei vom Altova Support erhalten haben, speichern Sie diese unter `<ProgramFilesFolder>\Altova\RaptorXMLServer2025\bin\`. Ihre DLL-Datei wird einen Namen in der Form `RaptorXMLServer_lc.dll` haben. Der `_lc` Teil des Namens enthält den Sprachencode. So steht z.B. in `RaptorXMLServer_de.dll` der `de` Teil für den Sprachencode für Deutsch.
5. Führen Sie den Befehl `setdeflang` [227] aus, um Ihre lokalisierte DLL als die zu verwendende RaptorXML Applikation zu definieren. Verwenden Sie den Sprachencode, der Teil des DLL-Namens ist, als Argument des Befehls `setdeflang` [227].

Anmerkung: Altova RaptorXML Server ist mit Unterstützung für fünf Sprachen erhältlich: Englisch, Deutsch, Spanisch, Französisch und Japanisch. Sie müssen daher keine lokalisierte Version dieser Sprachen erstellen. Um eine dieser fünf Sprachen als Standardsprache festzulegen, verwenden Sie den Befehl `setdeflang` [227].

5.8.1 exportresourcestrings

Syntax und Beschreibung

Der Befehl **`exportresourcestrings`** gibt eine XML-Datei aus, die die Ressourcenstrings der RaptorXML Server-Applikation in der definierten Sprache enthält. Als Exportsprachen stehen Englisch (`en`), Deutsch (`de`), Spanisch (`es`), Französisch (`fr`) und Japanisch (`ja`) zur Verfügung.

```
raptorxml exportresourcestrings [options] LanguageCode XMLOutputFile
raptorxmlserver exportresourcestrings [options] LanguageCode XMLOutputFile
```

- Das Argument ***`LanguageCode`*** gibt die Sprache der Ressourcenstrings in der XML-Ausgabedatei an; dies ist die *Exportsprache*. Derzeit unterstützte Exportsprachen sind (mit den Sprachcodes in Klammern): Englisch (`en`), Deutsch, (`de`), Spanisch (`es`), Französisch (`fr`) und Japanisch (`ja`).
- Das Argument ***`XMLOutputFile`*** definiert den Namen und Pfad der XML-Ausgabedatei.
- Der Befehl **`assignlicense`** kann von beiden ausführbaren Dateien aus aufgerufen werden: **`raptorxml`** und **`raptorxmlserver`**.

Eine Anleitung zum Erstellen von Lokalisierungen finden Sie weiter unten.

- Groß- und Kleinschreibung und Schrägstriche in der Befehlszeile

  `RaptorXML` (und **`RaptorXMLServer`** für Administrator-Befehle) *unter Windows*
  `raptorxml` und **`raptorxmlserver`** für Administrator-Befehle) *unter Windows und Unix (Linux, Mac)*

  * Beachten Sie, dass klein geschriebene Befehle (`raptorxml` und `raptorxmlserver`) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (`RaptorXML`) nur unter Windows und Mac ausgeführt werden.
  *Verwenden Sie auf Linux und Mac-Systemen Schrägstriche und auf Windows-Systemen umgekehrte Schrägstriche.

- Umgekehrte Schrägstriche, Leerzeichen und Sonderzeichen auf Windows-Systemen

  Auf Windows-Systemen: Verwenden Sie bei Vorhandensein von Leerzeichen oder Sonerzeichen in Strings (z.B. in Datei- oder Ordnernamen oder Firmen-, Personen- oder Produktnamen) Anführungszeichen: z.B., **`"Meine Datei"`**. Beachten Sie jedoch, dass ein von einem doppelten Anführungszeichen gefolgter umgekehrter Schrägstrich (z.B: **`"C:\Mein Verzeichnis\"`**) eventuell nicht korrekt gelesen wird, da der umgekehrte Schrägstrich auch den Beginn einer Escape-Sequenz markiert und die Escape-Sequenz **`\"`** für ein doppeltes Anführungszeichen steht. Wenn Sie diese Zeichensequenz mit einem Escape versehen wollen, verwenden Sie einen vorangestellten umgekehrten Schrägstrich, wie den folgenden: **`\\"`**. Einfacher ausgedrückt: Wenn Sie einen Dateipfad, der Leerzeichen oder einen umgekehrten Schrägstrich am Ende enthält, schreiben müssen, so schreiben Sie diesen folgendermaßen: **`"C:\Mein Verzeichnis\\"`**.

Beispiele

Beispiele für den Befehl **`exportresourcestrings`**:

```
raptorxml exportresourcestrings en c:\Strings.xml
raptorxmlserver exportresourcestrings en c:\Strings.xml
```

- Mit dem ersten Befehl oben wird unter `c:\` eine Datei namens `Strings.xml` erstellt, die alle Ressourcenstrings der RaptorXML Server Applikation in englischer Sprache enthält.
- Mit dem zweiten Befehl wird die ausführbare Server-Datei aufgerufen, um dasselbe zu tun, wie im ersten Beispiel.

Erstellen lokalisierter Versionen von RaptorXML Server

Sie können für jede Sprache Ihrer Wahl eine lokalisierte Version von RaptorXML Server erstellen. Im Ordner `C:\Programme (x86)\Altova\RaptorXMLServer2025\bin` stehen fünf lokalisierte Versionen (Englisch, Deutsch, Spanisch, Französisch und Japanisch) bereits zur Verfügung. Eine Lokalisierung für diese Sprache ist daher nicht mehr notwendig.

Folgendermaßen können Sie eine lokalisierte Version erstellen:

1. Generieren Sie mit Hilfe des Befehls `exportresourcestrings` (*siehe Befehlssyntax oben*) eine XML-Datei, die die Ressourcenstrings enthält. Die Ressourcenstrings in dieser XML-Datei sind in einer der fünf unterstützten Sprachen: je nachdem, welches ***`LanguageCode`***-Argument mit dem Befehl verwendet wird, in Englisch (`en`), Deutsch (`de`), Spanisch (`es`), Französisch (`fr`) oder Japanisch (`ja`).
2. Übersetzen Sie die Ressourcenstrings aus einer der fünf unterstützten Sprachen in die Zielsprache. Die Ressourcenstrings bilden den Inhalt der `<string>` Elemente in der XML-Datei. Übersetzen Sie

keine Variablen in geschweiften Klammern wie z.B. `{option}` oder `{product}`.

3. Wenden Sie sich an den Altova Support, um anhand Ihrer übersetzten XML-Datei eine lokalisierte RaptorXML Server DLL-Datei zu generieren.
4. Nachdem Sie Ihre lokalisierte DLL-Datei vom Altova Support erhalten haben, speichern Sie diese unter `C:\Programme (x86)\Altova\RaptorXMLServer2025\bin`. Ihre DLL-Datei wird einen Namen in der Form `RaptorXML2025_lc.dll` haben. Der `lc` Teil des Namens enthält den Sprachencode. So steht z.B. in `RaptorXML2025_de.dll` der Teil `de` für Deutsch.
5. Führen Sie den Befehl `setdeflang` aus, um Ihre lokalisierte DLL als die zu verwendende RaptorXML Server Applikation zu definieren. Verwenden Sie den Sprachencode, der Teil des DLL-Namens ist, als Argument des Befehls `setdeflang`.

Anmerkung: Altova RaptorXML Server ist mit Unterstützung für fünf Sprachen erhältlich: Englisch, Deutsch, Spanisch, Französisch und Japanisch. Sie müssen daher keine lokalisierte Version dieser Sprachen erstellen. Um eine dieser Sprachen als Standardsprache festzulegen, verwenden Sie den RaptorXML Server Befehl `setdeflang`.

5.8.2 setdeflang

Syntax und Beschreibung

Der Befehl **`setdeflang`** (Kurzform ist **`sdl`**) definiert die Standardsprache von RaptorXML Server. Verfügbare Sprachen sind Englisch (`en`), Deutsch (`de`), Spanisch (`es`), Französisch (`fr`) und Japanisch (`ja`). Der Befehl erhält ein obligatorisches Argument *`LanguageCode`*.

`raptorxml setdeflang [options]` *LanguageCode*
`raptorxmlserver setdeflang [options]` *LanguageCode*

- Das Argument *`LanguageCode`* definiert die Standardsprache von RaptorXML Server. Die entsprechenden Werte sind: `en`, `de`, `es`, `fr`, `ja`.
- Der Befehl **`setdeflang`** kann von beiden ausführbaren Dateien aus aufgerufen werden: **`raptorxml`** und **`raptorxmlserver`**.
- Mit Hilfe der Option **`--h, --help`** können Sie Informationen über den Befehl anzeigen.

▼ Groß- und Kleinschreibung und Schrägstriche in der Befehlszeile

`RaptorXML` (und **`RaptorXMLServer`** für Administrator-Befehle) *unter Windows*
`raptorxml` und **`raptorxmlserver`** für Administrator-Befehle) *unter Windows und Unix (Linux, Mac)*

* Beachten Sie, dass klein geschriebene Befehle (`raptorxml` und `raptorxmlserver`) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (`RaptorXML`) nur unter Windows und Mac ausgeführt werden.
*Verwenden Sie auf Linux und Mac-Systemen Schrägstriche und auf Windows-Systemen umgekehrte Schrägstriche.

Beispiele

Beispiele für den Befehl **`setdeflang`** (**`sdl`**):

```
raptorxml sdl en
raptorxml setdeflang es
```

```
raptorxmlserver setdeflang es
```

- Mit dem ersten Befehl wird als Standardsprache von RaptorXML Server Englisch definiert.
- Mit dem zweiten Befehl wird als Standardsprache von RaptorXML Server Spanisch definiert.
- Der dritte Befehl ist derselbe wie Befehl wie der zweite, wird jedoch von der ausführbaren Server-Datei ausgeführt.

Optionen

Mit Hilfe der Option `--h, --help` können Sie Informationen über den Befehl anzeigen.

5.9 Lizenzbefehle

In diesem Abschnitt sind Befehle für die Lizenzierung von RaptorXML Server beschrieben:

- licenseserver[229]: zum Registrieren von RaptorXML Server auf dem Altova LicenseServer in Ihrem Netzwerk
- assignlicense[230]: Um eine Lizenzdatei auf LicenseServer hochzuladen (nur Windows)
- verifylicense[232]: um zu überprüfen, ob RaptorXML Server lizenziert ist (nur Windows)

Anmerkung: Diese Befehle können auch über die ausführbare Serverdatei für Verwaltungsbefehle ausgeführt werden.[234]

Nähere Informationen zur Lizenzierung von Altova-Produkten mit Altova LicenseServer finden Sie in der Dokumentation zu Altova LicenseServer.

5.9.1 licenseserver

Syntax und Beschreibung

Bei Ausführung des Befehls **`licenseserver`** wird RaptorXML Server auf dem durch das Argument *`Server-Or-IP-Address`* definierten LicenseServer registriert. Damit der Befehl `licenseserver` erfolgreich ausgeführt werden kann, müssen sich die beiden Server (RaptorXML Server und LicenseServer) im selben Netzwerk befinden und LicenseServer muss ausgeführt werden. Außerdem benötigen Sie zum Registrieren von RaptorXML Server auf dem LicenseServer Administratorrechte.

`raptorxml licenseserver [options]` *Server-Or-IP-Address*
`raptorxmlserver licenseserver [options]` *Server-Or-IP-Address*

- Das Argument *`Server-Or-IP-Address`* erhält den Namen oder die IP-Adresse des LicenseServer-Rechners.
- Der Befehl **`licenseserver`** kann von beiden ausführbaren Dateien aus aufgerufen werden: **`raptorxml`** und **`raptorxmlserver`**.

Sobald RaptorXML Server erfolgreich auf dem LicenseServer registriert wurde, erhalten Sie eine entsprechende Meldung. Darin wird auch die URL des LicenseServer angezeigt. Sie können nun zu LicenseServer wechseln und RaptorXML Server eine Lizenz zuweisen. Nähere Informationen zur Lizenzierung finden Sie in der Dokumentation zu LicenseServer (https://www.altova.com/manual/de/licenseserver/3.17/).

▼ Groß- und Kleinschreibung und Schrägstriche in der Befehlszeile

`RaptorXML` (und **`RaptorXMLServer`** für Administrator-Befehle) *unter Windows*
`raptorxml` und **`raptorxmlserver`** für Administrator-Befehle) *unter Windows und Unix (Linux, Mac)*

* Beachten Sie, dass klein geschriebene Befehle (`raptorxml` und `raptorxmlserver`) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (`RaptorXML`) nur unter Windows und Mac ausgeführt werden.
*Verwenden Sie auf Linux und Mac-Systemen Schrägstriche und auf Windows-Systemen umgekehrte Schrägstriche.

▼ Umgekehrte Schrägstriche, Leerzeichen und Sonderzeichen auf Windows-Systemen

Auf Windows-Systemen: Verwenden Sie bei Vorhandensein von Leerzeichen oder Sonerzeichen in Strings (z.B. in Datei- oder Ordnernamen oder Firmen-, Personen- oder Produktnamen) Anführungszeichen: z.B., **"Meine Datei"**. Beachten Sie jedoch, dass ein von einem doppelten Anführungszeichen gefolgter umgekehrter Schrägstrich (z.B: **"C:\Mein Verzeichnis\"**) eventuell nicht korrekt gelesen wird, da der umgekehrte Schrägstrich auch den Beginn einer Escape-Sequenz markiert und die Escape-Sequenz **\"** für ein doppeltes Anführungszeichen steht. Wenn Sie diese Zeichensequenz mit einem Escape versehen wollen, verwenden Sie einen vorangestellten umgekehrten Schrägstrich, wie den folgenden: **\\"**. Einfacher ausgedrückt: Wenn Sie einen Dateipfad, der Leerzeichen oder einen umgekehrten Schrägstrich am Ende enthält, schreiben müssen, so schreiben Sie diesen folgendermaßen: **"C:\Mein Verzeichnis\\"**.

Beispiele

Beispiele für den Befehl **licenseserver**:

```
raptorxml licenseserver DOC.altova.com
raptorxml licenseserver localhost
raptorxml licenseserver 127.0.0.1
raptorxmlserver licenseserver 127.0.0.1
```

Die Befehle oben definieren den Rechner namens `DOC.altova.com` und den Rechner des Benutzers (`localhost` bzw. `127.0.0.1`) als den Rechner, auf dem Altova LicenseServer ausgeführt wird. In jedem dieser Fälle wird RaptorXML Server auf dem LicenseServer auf dem angegebenen Rechner registriert. Mit dem letzten Befehl wird zum Ausführen des Befehls die ausführbare Server-Datei aufgerufen.

Optionen

Optionen werden in ihrer kurzen Form (falls verfügbar) und in ihrer langen Form aufgelistet. Für die kurze und die lange Form können ein oder zwei Bindestriche verwendet werden. Eine Option kann, muss aber keinen Wert erhalten. Eine Option, die einen Wert erhält, wird folgendermaßen geschrieben: **--option=wert**. Werte können außer in zwei Fällen ohne Anführungszeichen definiert werden: (i) wenn der Wertestring Leerzeichen enthält oder (ii) wenn in der Beschreibung der Option explizit erwähnt ist, dass Anführungszeichen zwingend erforderlich sind. Wenn eine Option einen Booleschen Wert erhält und kein Wert definiert ist, so ist der Standardwert der Option `TRUE`. Mit Hilfe der Option **--h, --help** können Sie Informationen über den Befehl anzeigen.

▼ json [j]

--j, --json = `true|false`

Die Werte sind `true|false`. Bei `true` wird das Ergebnis des Registrierungsversuchs als JSON-Objekt ausgegeben, das von Rechnern geparst werden kann.

5.9.2 assignlicense (nur Windows)

Syntax und Beschreibung

Mit dem Befehl **assignlicense** wird eine Lizenzdatei auf den Altova LicenseServer, auf dem RaptorXML Server registriert ist (siehe Befehl `licenseserver`) hochgeladen und die Lizenz wird RaptorXML Server zugewiesen. Der Befehl erhält den Pfad einer Lizenzdatei als Argument. Außerdem können Sie mit dem Befehl die

Gültigkeit einer Lizenz überprüfen.

```
raptorxml assignlicense [options] FILE
raptorxmlserver assignlicense [options] FILE
```

- Das Argument *FILE* erhält den Pfad der Lizenzdatei.
- Mit der Option `--test-only` wird die Lizenzdatei auf LicenseServer hochgeladen und auf ihre Gültigkeit überprüft. Sie wird jedoch RaptorXML Server nicht zugewiesen.
- Der Befehl **assignlicense** kann von beiden ausführbaren Dateien aus aufgerufen werden: **raptorxml** und **raptorxmlserver**.

Nähere Informationen zur Lizenzierung finden Sie in der Dokumentation zu LicenseServer (https://www.altova.com/manual/de/licenseserver/3.17/).

▼ Groß- und Kleinschreibung und Schrägstriche in der Befehlszeile

RaptorXML (und **RaptorXMLServer** für Administrator-Befehle) *unter Windows*
raptorxml und **raptorxmlserver** für Administrator-Befehle) *unter Windows und Unix (Linux, Mac)*

* Beachten Sie, dass klein geschriebene Befehle (`raptorxml` und `raptorxmlserver`) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (`RaptorXML`) nur unter Windows und Mac ausgeführt werden.
*Verwenden Sie auf Linux und Mac-Systemen Schrägstriche und auf Windows-Systemen umgekehrte Schrägstriche.

▼ Umgekehrte Schrägstriche, Leerzeichen und Sonderzeichen auf Windows-Systemen

Auf Windows-Systemen: Verwenden Sie bei Vorhandensein von Leerzeichen oder Sonerzeichen in Strings (z.B. in Datei- oder Ordnernamen oder Firmen-, Personen- oder Produktnamen) Anführungszeichen: z.B., **"Meine Datei"**. Beachten Sie jedoch, dass ein von einem doppelten Anführungszeichen gefolgter umgekehrter Schrägstrich (z.B: **"C:\Mein Verzeichnis\"**) eventuell nicht korrekt gelesen wird, da der umgekehrte Schrägstrich auch den Beginn einer Escape-Sequenz markiert und die Escape-Sequenz **\"** für ein doppeltes Anführungszeichen steht. Wenn Sie diese Zeichensequenz mit einem Escape versehen wollen, verwenden Sie einen vorangestellten umgekehrten Schrägstrich, wie den folgenden: **\\"**. Einfacher ausgedrückt: Wenn Sie einen Dateipfad, der Leerzeichen oder einen umgekehrten Schrägstrich am Ende enthält, schreiben müssen, so schreiben Sie diesen folgendermaßen: **"C:\Mein Verzeichnis\\"**.

Beispiele

Beispiele für den Befehl `assignlicense`:

```
raptorxml assignlicense C:\licensepool\mylicensekey.altova_licenses
raptorxmlserver assignlicense C:\licensepool\mylicensekey.altova_licenses
raptorxml assignlicense --test-only=true C:\licensepool\mylicensekey.altova_licenses
```

- Mit dem ersten Befehl oben wird die angegebene Lizenzdatei auf LicenseServer hochgeladen und RaptorXML Server zugewiesen.
- Mit dem zweiten Befehl wird die ausführbare Server-Datei aufgerufen, um dasselbe zu tun, wie der erste Befehl.
- Mit dem letzten Befehl wird die angegebene Lizenz auf LicenseServer hochgeladen und auf ihre Gültigkeit überprüft, ohne sie RaptorXML Server zuzuweisen.

Optionen

Optionen werden in ihrer kurzen Form (falls verfügbar) und in ihrer langen Form aufgelistet. Für die kurze und die lange Form können ein oder zwei Bindestriche verwendet werden. Eine Option kann, muss aber keinen Wert erhalten. Eine Option, die einen Wert erhält, wird folgendermaßen geschrieben: **`--option=wert`**. Werte können außer in zwei Fällen ohne Anführungszeichen definiert werden: (i) wenn der Wertestring Leerzeichen enthält oder (ii) wenn in der Beschreibung der Option explizit erwähnt ist, dass Anführungszeichen zwingend erforderlich sind. Wenn eine Option einen Booleschen Wert erhält und kein Wert definiert ist, so ist der Standardwert der Option `TRUE`. Mit Hilfe der Option **`--h, --help`** können Sie Informationen über den Befehl anzeigen.

▼ test-only [t]

`--t, --test-only =` `true|false`
Die Werte sind `true|false`. Bei `true` wird die Lizenzdatei auf LicenseServer hochgeladen und auf ihre Gültigkeit überprüft, aber nicht zugewiesen.

5.9.3 verifylicense (nur Windows)

Syntax und Beschreibung

Mit dem Befehl **`verifylicense`** wird überprüft, ob das aktuelle Produkt lizenziert ist. Zusätzlich können Sie mit der Option `--license-key` überprüfen, ob dem Produkt bereits ein bestimmter Lizenzschlüssel zugewiesen wurde.

```
raptorxml verifylicense [options]
raptorxmlserver verifylicense [options]
```

- Um zu überprüfen, ob RaptorXML Server eine bestimmte Lizenz zugewiesen ist, geben Sie den Lizenzschlüssel als Wert der Option `--license-key` an.
- Der Befehl **`verifylicense`** kann von beiden ausführbaren Dateien aus aufgerufen werden: **`raptorxml`** und **`raptorxmlserver`**.

Nähere Informationen zur Lizenzierung finden Sie in der Dokumentation zu LicenseServer (https://www.altova.com/manual/de/licenseserver/3.17/).

▼ Groß- und Kleinschreibung und Schrägstriche in der Befehlszeile

`RaptorXML` (und **`RaptorXMLServer`** für Administrator-Befehle) *unter Windows*
`raptorxml` und **`raptorxmlserver`** für Administrator-Befehle) *unter Windows und Unix (Linux, Mac)*

* Beachten Sie, dass klein geschriebene Befehle (`raptorxml` und `raptorxmlserver`) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (`RaptorXML`) nur unter Windows und Mac ausgeführt werden.
*Verwenden Sie auf Linux und Mac-Systemen Schrägstriche und auf Windows-Systemen umgekehrte Schrägstriche.

Beispiele

Beispiele für den Befehl **verifylicense**:

```
raptorxml verifylicense
raptorxml verifylicense --license-key=ABCD123-ABCD123-ABCD123-ABCD123-ABCD123-ABCD123
raptorxmlserver verifylicense --license-key=ABCD123-ABCD123-ABCD123-ABCD123-ABCD123-
ABCD123
```

- Mit dem ersten Befehl wird überprüft, ob RaptorXML Server lizenziert ist.
- Mit dem zweiten Befehl wird überprüft, ob RaptorXML Server mit dem in der Option `--license-key` definierten Lizenzschlüssel lizenziert ist.
- Der dritte Befehl ist derselbe wie Befehl wie der zweite, wird jedoch von der ausführbaren Server-Datei ausgeführt.

Optionen

Optionen werden in ihrer kurzen Form (falls verfügbar) und in ihrer langen Form aufgelistet. Für die kurze und die lange Form können ein oder zwei Bindestriche verwendet werden. Eine Option kann, muss aber keinen Wert erhalten. Eine Option, die einen Wert erhält, wird folgendermaßen geschrieben: **--option=wert**. Werte können außer in zwei Fällen ohne Anführungszeichen definiert werden: (i) wenn der Wertestring Leerzeichen enthält oder (ii) wenn in der Beschreibung der Option explizit erwähnt ist, dass Anführungszeichen zwingend erforderlich sind. Wenn eine Option einen Booleschen Wert erhält und kein Wert definiert ist, so ist der Standardwert der Option `TRUE`. Mit Hilfe der Option **--h, --help** können Sie Informationen über den Befehl anzeigen.

▼ license-key [l]

--l, --license-key = *Value*

Überprüft, ob RaptorXML Server mit dem als Wert dieser Option definierten Lizenzschlüssel lizenziert ist.

5.10 Verwaltungsbefehle

Die Verwaltungsbefehle (wie z.B. installation-as-service und Lizenzierungsbefehle) werden an die ausführbare Serverdatei von RaptorXML Server (namens **RaptorXMLServer**) ausgegeben. Diese ausführbare Datei befindet sich standardmäßig unter dem folgenden Pfad:

Windows	`<ProgramFilesFolder>\Altova\RaptorXMLServer2025\bin\`**RaptorXMLServer.exe**
Linux	`/opt/Altova/RaptorXMLServer2025/bin/`**raptorxmlserver**
Mac	`/usr/local/Altova/RaptorXMLServer2025/bin/`**raptorxmlserver**

Verwendung

Die Befehlszeilensyntax lautet:

```
raptorxmlserver --h | --help | --version | <command> [options] [arguments]
```

- `--help` (Kurzform `--h`) Zeigt den Hilfetext zum jeweiligen Befehl an. Wenn kein Befehl angegeben ist, werden alle Befehle der ausführbaren Datei mit jeweils einer kurzen Beschreibung des Befehls aufgelistet.
- `--version` Zeigt die Versionsnummer von RaptorXML Server an.
- `<command>` ist der auszuführende Befehl. Die Befehle sind in den Unterabschnitten dieses Abschnitts beschrieben (*siehe Liste unten*).
- `[options]` sind die Optionen eines Befehls. Diese werden mit ihren jeweiligen Befehlen aufgelistet und beschrieben.
- `[arguments]` sind die Argumente eines Befehls. Diese werden mit ihren jeweiligen Befehlen aufgelistet und beschrieben.

▼ Groß- und Kleinschreibung und Schrägstriche in der Befehlszeile

RaptorXML (und **RaptorXMLServer** für Administrator-Befehle) *unter Windows*
raptorxml und **raptorxmlserver** für Administrator-Befehle) *unter Windows und Unix (Linux, Mac)*

* Beachten Sie, dass klein geschriebene Befehle (`raptorxml` und `raptorxmlserver`) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (`RaptorXML`) nur unter Windows und Mac ausgeführt werden.
*Verwenden Sie auf Linux und Mac-Systemen Schrägstriche und auf Windows-Systemen umgekehrte Schrägstriche.

Verwaltungsbefehle

Die Befehle der ausführbaren Server-Datei bieten Verwaltungsfunktionen. Im Folgenden finden Sie eine Liste der Befehle. Eine Beschreibung dazu folgt in den Unterabschnitten dieses Abschnitts:

5.10.1 install

Syntax und Beschreibung

Mit dem Befehl `install` wird RaptorXML Server als Dienst auf dem Server-Rechner installiert.

```
raptorxmlserver install [options]
```

- Beachten Sie, dass RaptorXML Server bei Installation als Dienst nicht automatisch gestartet wird. Um den Dienst zu starten, verwenden Sie den Befehl `start`.
- Um RaptorXML Server als Dienst zu deinstallieren, verwenden Sie den Befehl `uninstall`.
- Mit Hilfe der Option **`--h, --help`** können Sie Informationen über den Befehl anzeigen.

▼ Groß- und Kleinschreibung und Schrägstriche in der Befehlszeile

`RaptorXML` (und **`RaptorXMLServer`** für Administrator-Befehle) *unter Windows*
`raptorxml` und **`raptorxmlserver`** für Administrator-Befehle) *unter Windows und Unix (Linux, Mac)*

* Beachten Sie, dass klein geschriebene Befehle (`raptorxml` und `raptorxmlserver`) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (`RaptorXML`) nur unter Windows und Mac ausgeführt werden.
*Verwenden Sie auf Linux und Mac-Systemen Schrägstriche und auf Windows-Systemen umgekehrte Schrägstriche.

Beispiel

Beispiel für den Befehl **`install`**:

```
raptorxmlserver install
```

5.10.2 uninstall

Syntax und Beschreibung

Mit dem Befehl un**`install`** wird RaptorXML Server als Dienst auf dem Server-Rechner deinstalliert.

```
raptorxmlserver uninstall [options]
```

Um RaptorXML Server wieder als Dienst zu installieren, verwenden Sie den Befehl **`install`**.

▼ Groß- und Kleinschreibung und Schrägstriche in der Befehlszeile

RaptorXML (und **RaptorXMLServer** für Administrator-Befehle) *unter Windows*
raptorxml und **raptorxmlserver** für Administrator-Befehle) *unter Windows und Unix (Linux, Mac)*

* Beachten Sie, dass klein geschriebene Befehle (raptorxml und raptorxmlserver) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (RaptorXML) nur unter Windows und Mac ausgeführt werden.
*Verwenden Sie auf Linux und Mac-Systemen Schrägstriche und auf Windows-Systemen umgekehrte Schrägstriche.

Beispiel

Beispiel für den Befehl **uninstall**:

```
raptorxmlserver uninstall
```

5.10.3 start

Syntax und Beschreibung

Mit dem Befehl **start** wird RaptorXML Server als Dienst auf dem Server-Rechner gestartet.

```
raptorxmlserver start [options]
```

- Wenn RaptorXML Server nicht als Dienst installiert wurde, können Sie dies vor dem Start mit dem Befehl **install** tun.
- Um RaptorXML Server als Dienst zu deinstallieren, verwenden Sie den Befehl **uninstall**.
- Mit Hilfe der Option **--h, --help** können Sie Informationen über den Befehl anzeigen.

▼ Groß- und Kleinschreibung und Schrägstriche in der Befehlszeile

RaptorXML (und **RaptorXMLServer** für Administrator-Befehle) *unter Windows*
raptorxml und **raptorxmlserver** für Administrator-Befehle) *unter Windows und Unix (Linux, Mac)*

* Beachten Sie, dass klein geschriebene Befehle (raptorxml und raptorxmlserver) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (RaptorXML) nur unter Windows und Mac ausgeführt werden.
*Verwenden Sie auf Linux und Mac-Systemen Schrägstriche und auf Windows-Systemen umgekehrte Schrägstriche.

▼ Umgekehrte Schrägstriche, Leerzeichen und Sonderzeichen auf Windows-Systemen

Auf Windows-Systemen: Verwenden Sie bei Vorhandensein von Leerzeichen oder Sonerzeichen in Strings (z.B. in Datei- oder Ordnernamen oder Firmen-, Personen- oder Produktnamen) Anführungszeichen: z.B., **"Meine Datei"**. Beachten Sie jedoch, dass ein von einem doppelten Anführungszeichen gefolgter umgekehrter Schrägstrich (z.B: **"C:\Mein Verzeichnis\"**) eventuell nicht korrekt gelesen wird, da der umgekehrte Schrägstrich auch den Beginn einer Escape-Sequenz markiert und die Escape-Sequenz **\"** für ein doppeltes Anführungszeichen steht. Wenn Sie diese Zeichensequenz mit einem Escape versehen wollen, verwenden Sie einen vorangestellten umgekehrten Schrägstrich, wie den folgenden: **\\"**. Einfacher ausgedrückt: Wenn Sie einen Dateipfad, der Leerzeichen oder einen umgekehrten Schrägstrich am Ende enthält, schreiben müssen, so schreiben Sie diesen folgendermaßen: **"C:\Mein Verzeichnis\\"**.

Beispiel

Beispiel für den Befehl **start**:

```
raptorxmlserver start
```

Optionen

Optionen werden in ihrer kurzen Form (falls verfügbar) und in ihrer langen Form aufgelistet. Für die kurze und die lange Form können ein oder zwei Bindestriche verwendet werden. Eine Option kann, muss aber keinen Wert erhalten. Eine Option, die einen Wert erhält, wird folgendermaßen geschrieben: **--option=wert**. Werte können außer in zwei Fällen ohne Anführungszeichen definiert werden: (i) wenn der Wertestring Leerzeichen enthält oder (ii) wenn in der Beschreibung der Option explizit erwähnt ist, dass Anführungszeichen zwingend erforderlich sind. Wenn eine Option einen Booleschen Wert erhält und kein Wert definiert ist, so ist der Standardwert der Option `TRUE`. Mit Hilfe der Option **--h, --help** können Sie Informationen über den Befehl anzeigen.

- config [c]

  --c, --config = *File*
  Definiert den Pfad zur Konfigurationsdatei.

- fork

  --fork = `true|false`
  Bietet die Möglichkeit, bei Verwendung des klassischen `init` auf Unix-Servern zu verzweigen. Der Standardwert ist `false`.

- port

  --port = *PortNumber*
  Die Port-Nummer der Debug-Instanz von RaptorXML Server.

5.10.4 setdeflang

Syntax und Beschreibung

Der Befehl **setdeflang** (Kurzform ist **sdl**) definiert die Standardsprache von RaptorXML Server. Verfügbare Sprachen sind Englisch (`en`), Deutsch (`de`), Spanisch (`es`), Französisch (`fr`) und Japanisch (`ja`). Der Befehl erhält ein obligatorisches Argument *LanguageCode*.

```
raptorxml setdeflang [options] LanguageCode
raptorxmlserver setdeflang [options] LanguageCode
```

- Das Argument *LanguageCode* definiert die Standardsprache von RaptorXML Server. Die entsprechenden Werte sind: `en`, `de`, `es`, `fr`, `ja`.

- Der Befehl **setdeflang** kann von beiden ausführbaren Dateien aus aufgerufen werden: **raptorxml** und **raptorxmlserver**.
- Mit Hilfe der Option **--h, --help** können Sie Informationen über den Befehl anzeigen.

▼ Groß- und Kleinschreibung und Schrägstriche in der Befehlszeile

RaptorXML (und **RaptorXMLServer** für Administrator-Befehle) *unter Windows*
raptorxml und **raptorxmlserver** für Administrator-Befehle) *unter Windows und Unix (Linux, Mac)*

* Beachten Sie, dass klein geschriebene Befehle (`raptorxml` und `raptorxmlserver`) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (`RaptorXML`) nur unter Windows und Mac ausgeführt werden.
*Verwenden Sie auf Linux und Mac-Systemen Schrägstriche und auf Windows-Systemen umgekehrte Schrägstriche.

Beispiele

Beispiele für den Befehl **setdeflang** (**sdl**):

```
raptorxml sdl en
raptorxml setdeflang es
raptorxmlserver setdeflang es
```

- Mit dem ersten Befehl wird als Standardsprache von RaptorXML Server Englisch definiert.
- Mit dem zweiten Befehl wird als Standardsprache von RaptorXML Server Spanisch definiert.
- Der dritte Befehl ist derselbe wie Befehl wie der zweite, wird jedoch von der ausführbaren Server-Datei ausgeführt.

Optionen

Mit Hilfe der Option **--h, --help** können Sie Informationen über den Befehl anzeigen.

5.10.5 licenseserver

Syntax und Beschreibung

Bei Ausführung des Befehls **licenseserver** wird RaptorXML Server auf dem durch das Argument *Server-Or-IP-Address* definierten LicenseServer registriert. Damit der Befehl `licenseserver` erfolgreich ausgeführt werden kann, müssen sich die beiden Server (RaptorXML Server und LicenseServer) im selben Netzwerk befinden und LicenseServer muss ausgeführt werden. Außerdem benötigen Sie zum Registrieren von RaptorXML Server auf dem LicenseServer Administratorrechte.

```
raptorxml licenseserver [options] Server-Or-IP-Address
raptorxmlserver licenseserver [options] Server-Or-IP-Address
```

- Das Argument *Server-Or-IP-Address* erhält den Namen oder die IP-Adresse des LicenseServer-Rechners.
- Der Befehl **licenseserver** kann von beiden ausführbaren Dateien aus aufgerufen werden: **raptorxml** und **raptorxmlserver**.

Sobald RaptorXML Server erfolgreich auf dem LicenseServer registriert wurde, erhalten Sie eine entsprechende Meldung. Darin wird auch die URL des LicenseServer angezeigt. Sie können nun zu LicenseServer wechseln und RaptorXML Server eine Lizenz zuweisen. Nähere Informationen zur Lizenzierung finden Sie in der Dokumentation zu LicenseServer (https://www.altova.com/manual/de/licenseserver/3.17/).

- Groß- und Kleinschreibung und Schrägstriche in der Befehlszeile

  `RaptorXML` (und **`RaptorXMLServer`** für Administrator-Befehle) *unter Windows*
  `raptorxml` und **`raptorxmlserver`** für Administrator-Befehle) *unter Windows und Unix (Linux, Mac)*

  * Beachten Sie, dass klein geschriebene Befehle (`raptorxml` und `raptorxmlserver`) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (`RaptorXML`) nur unter Windows und Mac ausgeführt werden.
  *Verwenden Sie auf Linux und Mac-Systemen Schrägstriche und auf Windows-Systemen umgekehrte Schrägstriche.

- Umgekehrte Schrägstriche, Leerzeichen und Sonderzeichen auf Windows-Systemen

  Auf Windows-Systemen: Verwenden Sie bei Vorhandensein von Leerzeichen oder Sonerzeichen in Strings (z.B. in Datei- oder Ordnernamen oder Firmen-, Personen- oder Produktnamen) Anführungszeichen: z.B., **`"Meine Datei"`**. Beachten Sie jedoch, dass ein von einem doppelten Anführungszeichen gefolgter umgekehrter Schrägstrich (z.B: **`"C:\Mein Verzeichnis\"`**) eventuell nicht korrekt gelesen wird, da der umgekehrte Schrägstrich auch den Beginn einer Escape-Sequenz markiert und die Escape-Sequenz **`\"`** für ein doppeltes Anführungszeichen steht. Wenn Sie diese Zeichensequenz mit einem Escape versehen wollen, verwenden Sie einen vorangestellten umgekehrten Schrägstrich, wie den folgenden: **`\\"`**. Einfacher ausgedrückt: Wenn Sie einen Dateipfad, der Leerzeichen oder einen umgekehrten Schrägstrich am Ende enthält, schreiben müssen, so schreiben Sie diesen folgendermaßen: **`"C:\Mein Verzeichnis\\"`**.

Beispiele

Beispiele für den Befehl **`licenseserver`**:

```
raptorxml licenseserver DOC.altova.com
raptorxml licenseserver localhost
raptorxml licenseserver 127.0.0.1
raptorxmlserver licenseserver 127.0.0.1
```

Die Befehle oben definieren den Rechner namens `DOC.altova.com` und den Rechner des Benutzers (`localhost` bzw. `127.0.0.1`) als den Rechner, auf dem Altova LicenseServer ausgeführt wird. In jedem dieser Fälle wird RaptorXML Server auf dem LicenseServer auf dem angegebenen Rechner registriert. Mit dem letzten Befehl wird zum Ausführen des Befehls die ausführbare Server-Datei aufgerufen.

Optionen

Optionen werden in ihrer kurzen Form (falls verfügbar) und in ihrer langen Form aufgelistet. Für die kurze und die lange Form können ein oder zwei Bindestriche verwendet werden. Eine Option kann, muss aber keinen Wert erhalten. Eine Option, die einen Wert erhält, wird folgendermaßen geschrieben: **`--option=wert`**. Werte können außer in zwei Fällen ohne Anführungszeichen definiert werden: (i) wenn der Wertestring Leerzeichen enthält oder (ii) wenn in der Beschreibung der Option explizit erwähnt ist, dass Anführungszeichen zwingend erforderlich sind. Wenn eine Option einen Booleschen Wert erhält und kein Wert definiert ist, so ist der Standardwert der Option `TRUE`. Mit Hilfe der Option **`--h, --help`** können Sie Informationen über den Befehl

anzeigen.

- json [j]

  --j, --json = `true|false`
  Die Werte sind `true|false`. Bei `true` wird das Ergebnis des Registrierungsversuchs als JSON-Objekt ausgegeben, das von Rechnern geparst werden kann.

5.10.6 accepteula (nur Linux)

Syntax und Beschreibung

Um RaptorXML Server ausführen zu können, muss die Endbenutzer-Lizenzvereinbarung (EULA) akzeptiert werden. Sie können die EULA der Applikation mit dem Befehl **accepteula** akzeptieren.

Dieser Befehl ist z.B. nützlich, wenn Sie RaptorXML Server direkt über automatisierte Prozesse unter Verwendung von Skripts lizenzieren und ausführen möchten.

```
raptorxml accepteula [options]
raptorxmlserver accepteula [options]
```

- Der Befehl funktioniert nur bei Altova Server-Produkten, die auf Linux-Rechnern installiert wurden.
- Sie müssen RaptorXML Server zuerst auf dem LicenseServer registrieren, bevor Sie den Befehl **accepteula** ausführen.
- Mit Hilfe der Option **--h, --help** können Sie Informationen über den Befehl anzeigen.
- Schreiben Sie den Befehl `raptorxml` und `raptorxmlserver` in Kleinbuchstaben.
- Verwenden Sie auf Linux-Systemen Vorwärts-Schrägstriche.

Beispiele

Beispiele für den Befehl **accepteula**:

```
raptorxml accepteula
raptorxmlserver accepteula
```

Optionen

Mit Hilfe der Option **--h, --help** können Sie Informationen über den Befehl anzeigen.

5.10.7 assignlicense

Syntax und Beschreibung

Mit dem Befehl **assignlicense** wird eine Lizenzdatei auf den Altova LicenseServer, auf dem RaptorXML Server registriert ist (siehe Befehl `licenseserver`) hochgeladen und die Lizenz wird RaptorXML Server zugewiesen. Der Befehl erhält den Pfad einer Lizenzdatei als Argument. Außerdem können Sie mit dem Befehl die Gültigkeit einer Lizenz überprüfen.

```
raptorxml assignlicense [options] FILE
raptorxmlserver assignlicense [options] FILE
```

- Das Argument *FILE* erhält den Pfad der Lizenzdatei.
- Mit der Option `--test-only` wird die Lizenzdatei auf LicenseServer hochgeladen und auf ihre Gültigkeit überprüft. Sie wird jedoch RaptorXML Server nicht zugewiesen.
- Der Befehl **assignlicense** kann von beiden ausführbaren Dateien aus aufgerufen werden: **raptorxml** und **raptorxmlserver**.

Nähere Informationen zur Lizenzierung finden Sie in der Dokumentation zu LicenseServer (https://www.altova.com/manual/de/licenseserver/3.17/).

▼ Groß- und Kleinschreibung und Schrägstriche in der Befehlszeile

RaptorXML (und **RaptorXMLServer** für Administrator-Befehle) *unter Windows*
raptorxml und **raptorxmlserver** für Administrator-Befehle) *unter Windows und Unix (Linux, Mac)*

* Beachten Sie, dass klein geschriebene Befehle (`raptorxml` und `raptorxmlserver`) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (`RaptorXML`) nur unter Windows und Mac ausgeführt werden.
*Verwenden Sie auf Linux und Mac-Systemen Schrägstriche und auf Windows-Systemen umgekehrte Schrägstriche.

▼ Umgekehrte Schrägstriche, Leerzeichen und Sonderzeichen auf Windows-Systemen

Auf Windows-Systemen: Verwenden Sie bei Vorhandensein von Leerzeichen oder Sonerzeichen in Strings (z.B. in Datei- oder Ordnernamen oder Firmen-, Personen- oder Produktnamen) Anführungszeichen: z.B., **"Meine Datei"**. Beachten Sie jedoch, dass ein von einem doppelten Anführungszeichen gefolgter umgekehrter Schrägstrich (z.B: **"C:\Mein Verzeichnis\"**) eventuell nicht korrekt gelesen wird, da der umgekehrte Schrägstrich auch den Beginn einer Escape-Sequenz markiert und die Escape-Sequenz **\"** für ein doppeltes Anführungszeichen steht. Wenn Sie diese Zeichensequenz mit einem Escape versehen wollen, verwenden Sie einen vorangestellten umgekehrten Schrägstrich, wie den folgenden: **\\"**. Einfacher ausgedrückt: Wenn Sie einen Dateipfad, der Leerzeichen oder einen umgekehrten Schrägstrich am Ende enthält, schreiben müssen, so schreiben Sie diesen folgendermaßen: **"C:\Mein Verzeichnis\\"**.

Beispiele

Beispiele für den Befehl `assignlicense`:

```
raptorxml assignlicense C:\licensepool\mylicensekey.altova_licenses
raptorxmlserver assignlicense C:\licensepool\mylicensekey.altova_licenses
raptorxml assignlicense --test-only=true C:\licensepool\mylicensekey.altova_licenses
```

- Mit dem ersten Befehl oben wird die angegebene Lizenzdatei auf LicenseServer hochgeladen und RaptorXML Server zugewiesen.
- Mit dem zweiten Befehl wird die ausführbare Server-Datei aufgerufen, um dasselbe zu tun, wie der erste Befehl.
- Mit dem letzten Befehl wird die angegebene Lizenz auf LicenseServer hochgeladen und auf ihre Gültigkeit überprüft, ohne sie RaptorXML Server zuzuweisen.

Optionen

Optionen werden in ihrer kurzen Form (falls verfügbar) und in ihrer langen Form aufgelistet. Für die kurze und die lange Form können ein oder zwei Bindestriche verwendet werden. Eine Option kann, muss aber keinen Wert erhalten. Eine Option, die einen Wert erhält, wird folgendermaßen geschrieben: **`--option=wert`**. Werte können außer in zwei Fällen ohne Anführungszeichen definiert werden: (i) wenn der Wertestring Leerzeichen enthält oder (ii) wenn in der Beschreibung der Option explizit erwähnt ist, dass Anführungszeichen zwingend erforderlich sind. Wenn eine Option einen Booleschen Wert erhält und kein Wert definiert ist, so ist der Standardwert der Option `TRUE`. Mit Hilfe der Option **`--h, --help`** können Sie Informationen über den Befehl anzeigen.

- test-only [t]

  `--t, --test-only =` `true|false`
  Die Werte sind `true|false`. Bei `true` wird die Lizenzdatei auf LicenseServer hochgeladen und auf ihre Gültigkeit überprüft, aber nicht zugewiesen.

5.10.8 verifylicense

Syntax und Beschreibung

Mit dem Befehl **`verifylicense`** wird überprüft, ob das aktuelle Produkt lizenziert ist. Zusätzlich können Sie mit der Option `--license-key` überprüfen, ob dem Produkt bereits ein bestimmter Lizenzschlüssel zugewiesen wurde.

```
raptorxml verifylicense [options]
raptorxmlserver verifylicense [options]
```

- Um zu überprüfen, ob RaptorXML Server eine bestimmte Lizenz zugewiesen ist, geben Sie den Lizenzschlüssel als Wert der Option `--license-key` an.
- Der Befehl **`verifylicense`** kann von beiden ausführbaren Dateien aus aufgerufen werden: **`raptorxml`** und **`raptorxmlserver`**.

Nähere Informationen zur Lizenzierung finden Sie in der Dokumentation zu LicenseServer (https://www.altova.com/manual/de/licenseserver/3.17/).

- Groß- und Kleinschreibung und Schrägstriche in der Befehlszeile

  `RaptorXML` (und **`RaptorXMLServer`** für Administrator-Befehle) *unter Windows*
  `raptorxml` und **`raptorxmlserver`** für Administrator-Befehle) *unter Windows und Unix (Linux, Mac)*

  * Beachten Sie, dass klein geschriebene Befehle (`raptorxml` und `raptorxmlserver`) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (`RaptorXML`) nur unter Windows und Mac ausgeführt werden.
  *Verwenden Sie auf Linux und Mac-Systemen Schrägstriche und auf Windows-Systemen umgekehrte Schrägstriche.

Beispiele

Beispiele für den Befehl **verifylicense**:

```
raptorxml verifylicense
raptorxml verifylicense --license-key=ABCD123-ABCD123-ABCD123-ABCD123-ABCD123-ABCD123
raptorxmlserver verifylicense --license-key=ABCD123-ABCD123-ABCD123-ABCD123-ABCD123-
ABCD123
```

- Mit dem ersten Befehl wird überprüft, ob RaptorXML Server lizenziert ist.
- Mit dem zweiten Befehl wird überprüft, ob RaptorXML Server mit dem in der Option `--license-key` definierten Lizenzschlüssel lizenziert ist.
- Der dritte Befehl ist derselbe wie Befehl wie der zweite, wird jedoch von der ausführbaren Server-Datei ausgeführt.

Optionen

Optionen werden in ihrer kurzen Form (falls verfügbar) und in ihrer langen Form aufgelistet. Für die kurze und die lange Form können ein oder zwei Bindestriche verwendet werden. Eine Option kann, muss aber keinen Wert erhalten. Eine Option, die einen Wert erhält, wird folgendermaßen geschrieben: **--option=wert**. Werte können außer in zwei Fällen ohne Anführungszeichen definiert werden: (i) wenn der Wertestring Leerzeichen enthält oder (ii) wenn in der Beschreibung der Option explizit erwähnt ist, dass Anführungszeichen zwingend erforderlich sind. Wenn eine Option einen Booleschen Wert erhält und kein Wert definiert ist, so ist der Standardwert der Option `TRUE`. Mit Hilfe der Option **--h, --help** können Sie Informationen über den Befehl anzeigen.

▼ license-key [l]

--l, --license-key = *Value*
Überprüft, ob RaptorXML Server mit dem als Wert dieser Option definierten Lizenzschlüssel lizenziert ist.

5.10.9 createconfig

Syntax und Beschreibung

Mit dem Befehl **createconfig** wird die Server-Konfigurationsdatei mit den Standardwerten überschrieben.

```
raptorxmlserver createconfig [options]
```

- Mit der Option **--lang** wird die Standardsprache der Server-Konfigurationsdatei definiert.

Nähere Informationen zu Server-Konfigurationsdatei finden Sie unter Konfigurieren des Servers 272.

▼ Groß- und Kleinschreibung und Schrägstriche in der Befehlszeile

RaptorXML (und **RaptorXMLServer** für Administrator-Befehle) *unter Windows*
raptorxml und **raptorxmlserver** für Administrator-Befehle) *unter Windows und Unix (Linux, Mac)*

* Beachten Sie, dass klein geschriebene Befehle (`raptorxml` und `raptorxmlserver`) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (`RaptorXML`) nur unter

Windows und Mac ausgeführt werden.
*Verwenden Sie auf Linux und Mac-Systemen Schrägstriche und auf Windows-Systemen umgekehrte Schrägstriche.

Beispiele

Beispiele für den Befehl **createconfig**:

```
raptorxml createconfig
raptorxml createconfig --lang=de
```

Optionen

- lang

  --lang = en|de|es|fr|ja
  Definiert die Standardsprache der Server-Konfigurationsdatei. Es stehen die folgenden Optionen zur Verfügung: Englisch (en), Deutsch (de), Spanisch (es), Französisch (fr), Japanisch (ja). Wenn die Option nicht definiert wird, wird als Standardsprache Englisch gewählt.

Mit Hilfe der Option **--h, --help** können Sie Informationen über den Befehl anzeigen.

Optionen werden in ihrer kurzen Form (falls verfügbar) und in ihrer langen Form aufgelistet. Für die kurze und die lange Form können ein oder zwei Bindestriche verwendet werden. Eine Option kann, muss aber keinen Wert erhalten. Eine Option, die einen Wert erhält, wird folgendermaßen geschrieben: **--option=wert**. Werte können außer in zwei Fällen ohne Anführungszeichen definiert werden: (i) wenn der Wertestring Leerzeichen enthält oder (ii) wenn in der Beschreibung der Option explizit erwähnt ist, dass Anführungszeichen zwingend erforderlich sind. Wenn eine Option einen Booleschen Wert erhält und kein Wert definiert ist, so ist der Standardwert der Option TRUE. Mit Hilfe der Option **--h, --help** können Sie Informationen über den Befehl anzeigen.

5.10.10 exportresourcestrings

Syntax und Beschreibung

Der Befehl **exportresourcestrings** gibt eine XML-Datei aus, die die Ressourcenstrings der RaptorXML Server-Applikation in der definierten Sprache enthält. Als Exportsprachen stehen Englisch (en), Deutsch (de), Spanisch (es), Französisch (fr) und Japanisch (ja) zur Verfügung.

```
raptorxml exportresourcestrings [options] LanguageCode XMLOutputFile
raptorxmlserver exportresourcestrings [options] LanguageCode XMLOutputFile
```

- Das Argument ***LanguageCode*** gibt die Sprache der Ressourcenstrings in der XML-Ausgabedatei an; dies ist die *Exportsprache*. Derzeit unterstützte Exportsprachen sind (mit den Sprachcodes in Klammern): Englisch (en), Deutsch, (de), Spanisch (es), Französisch (fr) und Japanisch (ja).
- Das Argument ***XMLOutputFile*** definiert den Namen und Pfad der XML-Ausgabedatei.
- Der Befehl **assignlicense** kann von beiden ausführbaren Dateien aus aufgerufen werden: **raptorxml**

und **raptorxmlserver**.

Eine Anleitung zum Erstellen von Lokalisierungen finden Sie weiter unten.

- Groß- und Kleinschreibung und Schrägstriche in der Befehlszeile

  RaptorXML (und **RaptorXMLServer** für Administrator-Befehle) *unter Windows*
  raptorxml und **raptorxmlserver** für Administrator-Befehle) *unter Windows und Unix (Linux, Mac)*

  * Beachten Sie, dass klein geschriebene Befehle (`raptorxml` und `raptorxmlserver`) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (`RaptorXML`) nur unter Windows und Mac ausgeführt werden.
  *Verwenden Sie auf Linux und Mac-Systemen Schrägstriche und auf Windows-Systemen umgekehrte Schrägstriche.

- Umgekehrte Schrägstriche, Leerzeichen und Sonderzeichen auf Windows-Systemen

  Auf Windows-Systemen: Verwenden Sie bei Vorhandensein von Leerzeichen oder Sonerzeichen in Strings (z.B. in Datei- oder Ordnernamen oder Firmen-, Personen- oder Produktnamen) Anführungszeichen: z.B., **"Meine Datei"**. Beachten Sie jedoch, dass ein von einem doppelten Anführungszeichen gefolgter umgekehrter Schrägstrich (z.B: **"C:\Mein Verzeichnis\"**) eventuell nicht korrekt gelesen wird, da der umgekehrte Schrägstrich auch den Beginn einer Escape-Sequenz markiert und die Escape-Sequenz **\"** für ein doppeltes Anführungszeichen steht. Wenn Sie diese Zeichensequenz mit einem Escape versehen wollen, verwenden Sie einen vorangestellten umgekehrten Schrägstrich, wie den folgenden: **\\"**. Einfacher ausgedrückt: Wenn Sie einen Dateipfad, der Leerzeichen oder einen umgekehrten Schrägstrich am Ende enthält, schreiben müssen, so schreiben Sie diesen folgendermaßen: **"C:\Mein Verzeichnis\\"**.

Beispiele

Beispiele für den Befehl **exportresourcestrings**:

```
raptorxml exportresourcestrings en c:\Strings.xml
raptorxmlserver exportresourcestrings en c:\Strings.xml
```

- Mit dem ersten Befehl oben wird unter `c:\` eine Datei namens `Strings.xml` erstellt, die alle Ressourcenstrings der RaptorXML Server Applikation in englischer Sprache enthält.
- Mit dem zweiten Befehl wird die ausführbare Server-Datei aufgerufen, um dasselbe zu tun, wie im ersten Beispiel.

Erstellen lokalisierter Versionen von RaptorXML Server

Sie können für jede Sprache Ihrer Wahl eine lokalisierte Version von RaptorXML Server erstellen. Im Ordner `C:\Programme (x86)\Altova\RaptorXMLServer2025\bin` stehen fünf lokalisierte Versionen (Englisch, Deutsch, Spanisch, Französisch und Japanisch) bereits zur Verfügung. Eine Lokalisierung für diese Sprache ist daher nicht mehr notwendig.

Folgendermaßen können Sie eine lokalisierte Version erstellen:

1. Generieren Sie mit Hilfe des Befehls `exportresourcestrings` (*siehe Befehlssyntax oben*) eine XML-Datei, die die Ressourcenstrings enthält. Die Ressourcenstrings in dieser XML-Datei sind in einer der fünf unterstützten Sprachen: je nachdem, welches ***LanguageCode***-Argument mit dem Befehl verwendet

wird, in Englisch (`en`), Deutsch (`de`), Spanisch (`es`), Französisch (`fr`) oder Japanisch (`ja`).
2. Übersetzen Sie die Ressourcenstrings aus einer der fünf unterstützten Sprachen in die Zielsprache. Die Ressourcenstrings bilden den Inhalt der `<string>` Elemente in der XML-Datei. Übersetzen Sie keine Variablen in geschweiften Klammern wie z.B. `{option}` oder `{product}`.
3. Wenden Sie sich an den Altova Support, um anhand Ihrer übersetzten XML-Datei eine lokalisierte RaptorXML Server DLL-Datei zu generieren.
4. Nachdem Sie Ihre lokalisierte DLL-Datei vom Altova Support erhalten haben, speichern Sie diese unter `C:\Programme (x86)\Altova\RaptorXMLServer2025\bin`. Ihre DLL-Datei wird einen Namen in der Form `RaptorXML2025_lc.dll` haben. Der `lc` Teil des Namens enthält den Sprachencode. So steht z.B. in `RaptorXML2025_de.dll` der Teil `de` für Deutsch.
5. Führen Sie den Befehl `setdeflang` aus, um Ihre lokalisierte DLL als die zu verwendende RaptorXML Server Applikation zu definieren. Verwenden Sie den Sprachencode, der Teil des DLL-Namens ist, als Argument des Befehls `setdeflang`.

Anmerkung: Altova RaptorXML Server ist mit Unterstützung für fünf Sprachen erhältlich: Englisch, Deutsch, Spanisch, Französisch und Japanisch. Sie müssen daher keine lokalisierte Version dieser Sprachen erstellen. Um eine dieser Sprachen als Standardsprache festzulegen, verwenden Sie den RaptorXML Server Befehl `setdeflang`.

5.10.11 debug

Syntax und Beschreibung

Mit dem Befehl **`debug`** wird RaptorXML Server für das Debuggen - und nicht als Dienst - gestartet. Um RaptorXML Server in diesem Modus zu beenden, drücken Sie **Strg+C**.

```
raptorxmlserver debug [options]
```

▼ Groß- und Kleinschreibung und Schrägstriche in der Befehlszeile

`RaptorXML` (und **`RaptorXMLServer`** für Administrator-Befehle) *unter Windows*
`raptorxml` und **`raptorxmlserver`** für Administrator-Befehle) *unter Windows und Unix (Linux, Mac)*

* Beachten Sie, dass klein geschriebene Befehle (`raptorxml` und `raptorxmlserver`) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (`RaptorXML`) nur unter Windows und Mac ausgeführt werden.
*Verwenden Sie auf Linux und Mac-Systemen Schrägstriche und auf Windows-Systemen umgekehrte Schrägstriche.

▼ Umgekehrte Schrägstriche, Leerzeichen und Sonderzeichen auf Windows-Systemen

Auf Windows-Systemen: Verwenden Sie bei Vorhandensein von Leerzeichen oder Sonerzeichen in Strings (z.B. in Datei- oder Ordnernamen oder Firmen-, Personen- oder Produktnamen) Anführungszeichen: z.B., **`"Meine Datei"`**. Beachten Sie jedoch, dass ein von einem doppelten Anführungszeichen gefolgter umgekehrter Schrägstrich (z.B: **`"C:\Mein Verzeichnis\"`**) eventuell nicht korrekt gelesen wird, da der umgekehrte Schrägstrich auch den Beginn einer Escape-Sequenz markiert und die Escape-Sequenz **`\"`** für ein doppeltes Anführungszeichen steht. Wenn Sie diese Zeichensequenz mit einem Escape versehen wollen, verwenden Sie einen vorangestellten umgekehrten Schrägstrich, wie den folgenden: **`\\"`**. Einfacher ausgedrückt: Wenn Sie einen Dateipfad, der Leerzeichen oder einen umgekehrten Schrägstrich am Ende enthält, schreiben müssen, so schreiben Sie diesen folgendermaßen: **`"C:\Mein Verzeichnis\\"`**.

Beispiel

Beispiel für den Befehl **debug**:

```
raptorxmlserver debug
```

Optionen

Optionen werden in ihrer kurzen Form (falls verfügbar) und in ihrer langen Form aufgelistet. Für die kurze und die lange Form können ein oder zwei Bindestriche verwendet werden. Eine Option kann, muss aber keinen Wert erhalten. Eine Option, die einen Wert erhält, wird folgendermaßen geschrieben: **--option=wert**. Werte können außer in zwei Fällen ohne Anführungszeichen definiert werden: (i) wenn der Wertestring Leerzeichen enthält oder (ii) wenn in der Beschreibung der Option explizit erwähnt ist, dass Anführungszeichen zwingend erforderlich sind. Wenn eine Option einen Booleschen Wert erhält und kein Wert definiert ist, so ist der Standardwert der Option TRUE. Mit Hilfe der Option **--h, --help** können Sie Informationen über den Befehl anzeigen.

- config [c]

  --c, --config = *File*
  Definiert den Pfad zu einer Konfigurationsdatei.

- port

  --port = *PortNumber*
  Die Port-Nummer der Debug-Instanz von RaptorXML Server

5.10.12 help

Syntax und Beschreibung

Der Befehl **help** hat ein einziges Argument (Command): den Namen des Befehls, zu dem die Hilfe benötigt wird. Er zeigt die korrekte Syntax des Befehls, seine Optionen sowie andere relevante Informationen an. Wenn das Argument Command nicht angegeben wird, werden alle Befehle der ausführbaren Datei aufgelistet, wobei zu jedem eine kurze Textbeschreibung angezeigt wird. Der Befehl **help** kann von beiden ausführbaren Dateien aus aufgerufen werden: **raptorxml** und **raptorxmlserver**.

```
raptorxml help Command
raptorxmlserver help Command
```

- Groß- und Kleinschreibung und Schrägstriche in der Befehlszeile

  RaptorXML (und **RaptorXMLServer** für Administrator-Befehle) *unter Windows*
  raptorxml und **raptorxmlserver** für Administrator-Befehle) *unter Windows und Unix (Linux, Mac)*

  * Beachten Sie, dass klein geschriebene Befehle (raptorxml und raptorxmlserver) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (RaptorXML) nur unter Windows und Mac ausgeführt werden.
  *Verwenden Sie auf Linux und Mac-Systemen Schrägstriche und auf Windows-Systemen umgekehrte

Schrägstriche.

Beispiel

Beispiele für den Befehl **help**, um Informationen über den Befehl **licenserver** (Dieser Befehl steht in beiden ausführbaren Dateien zur Verfügung) anzuzeigen:

```
raptorxml help licenseserver
raptorxmlserver help licenseserver
```

Die Option --help

Die Hilfe zu einem Befehl kann auch über die Option `--help` im Anschluss an diesen Befehl aufgerufen werden. Mit den beiden unten stehenden Befehlen erhalten Sie dasselbe Ergebnis:

```
raptorxml licenseserver --help
```

Im obigen Befehl wird die Option `--help` des Befehls **licenseserver** verwendet.

```
raptorxml help licenseserver
```

Der Befehl **help** erhält `licenseserver` als Argument.

In beiden Fällen wird die Hilfe zum Befehl **licenseserver** angezeigt.

5.10.13 version

Syntax und Beschreibung

Mit dem Befehl **version** wird die Versionsnummer von RaptorXML Server angezeigt. Der Befehl kann sowohl von der ausführbaren **raptorxml**-Datei als auch von der ausführbaren **raptorxmlserver**-Datei aus aufgerufen werden.

```
raptorxml version
raptorxmlserver version
```

▼ Groß- und Kleinschreibung und Schrägstriche in der Befehlszeile

RaptorXML (und **RaptorXMLServer** für Administrator-Befehle) *unter Windows*
raptorxml und **raptorxmlserver** für Administrator-Befehle) *unter Windows und Unix (Linux, Mac)*

* Beachten Sie, dass klein geschriebene Befehle (`raptorxml` und `raptorxmlserver`) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (`RaptorXML`) nur unter Windows und Mac ausgeführt werden.
*Verwenden Sie auf Linux und Mac-Systemen Schrägstriche und auf Windows-Systemen umgekehrte Schrägstriche.

Beispiel

Beispiele für den Befehl `version`:

```
raptorxml version
raptorxmlserver version
```

5.11 Optionen

Dieser Abschnitt enthält eine Beschreibung aller CLI-Optionen, geordnet nach Funktionalität. Informationen darüber, welche Optionen mit den einzelnen Befehlen verwendet werden können, finden Sie in der Beschreibung zu den entsprechenden Befehlen.

5.11.1 Kataloge, globale Ressourcen, ZIP-Dateien

▼ catalog

--catalog = *FILE*
Gibt den absoluten Pfad zu einer Root-Katalog-Datei an, die nicht die installierte Root-Katalog-Datei ist. Der Standardwert ist der absolute Pfad zur installierten Root-Katalog-Datei. (`<installationsordner>\Altova\RaptorXMLServer2025\etc\RootCatalog.xml`). Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge 48.

▼ user-catalog

--user-catalog = *FILE*
Definiert den absoluten Pfad zu einem XML-Katalog, der zusätzlich zum Root-Katalog verwendet werden soll. Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge 48.

▼ enable-globalresources

--enable-globalresources = `true|false`
Aktiviert die globalen Ressourcen 55. Standardwert ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

▼ globalresourceconfig [gc]

--gc | --globalresourceconfig = *VALUE*
Definiert die aktive Konfiguration der globalen Ressource 55 (und aktiviert globale Ressourcen) 55.

▼ globalresourcefile [gr]

--gr | --globalresourcefile = *FILE*
Definiert die globale Ressourcendatei 55 (und aktiviert globale Ressourcen) 55.

▼ recurse

--recurse = `true|false`
Dient zur Auswahl von Dateien innerhalb von Unterverzeichnissen einschließlich ZIP-Archiven. Bei `true` wählt das Argument *InputFile* des Befehls die angegebene Datei auch in den Unterverzeichnissen aus. Beispiel: `"test.zip|zip\test.xml"` wählt Dateien mit dem Namen `test.xml` auf allen Ordnerebenen des ZIP-Ordners aus. Referenzen auf ZIP-Dateien müssen in Anführungszeichen angegeben werden. Es können die Platzhalter `*` und `?` verwendet werden. Mit `*.xml` werden folglich alle `.xml` Dateien im (ZIP-) Ordner ausgewählt. Der Standardwert der Option ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

5.11.2 Meldungen, Fehler, Hilfe, Timeout, Version

error-format

--error-format = `text|shortxml|longxml`
Definiert das Format der Fehlerausgabe. Der Standardwert ist `text`. Mit den anderen Optionen werden XML-Formate generiert, wobei mit `longxml` mehr Details generiert werden.

error-limit

--error-limit = *N | unlimited*
Definiert das Fehlerlimit mit einem Wertebereich von `1` bis `9999` oder `unbegrenzt`. Der Standardwert ist `100`. Bei Erreichung des Fehlerlimits wird die Validierung gestoppt. Dient dazu, die Prozessorverwendung während der Validierung/Transformation einzuschränken.

help

--help
Zeigt den Hilfetext zum Befehl an. Beispiel: `valany --h`. (Alternativ dazu kann der Befehl `help` zusammen mit einem Argument verwendet werden. Beispiel: `help valany`.)

log-output

--log-output = *FILE*
Schreibt die Meldungsausgabe in die angegebene URL. Stellen Sie sicher, dass das CLI Schreibrechte für den Ausgabepfad hat.

network-timeout

--network-timeout = *VALUE*
Definiert das Timeout für entfernte I/O-Operationen in Millisekunden. Der Standardwert ist: `40000`.

verbose

--verbose = `true|false`
Mit dem Wert `true` wird die Ausgabe zusätzlicher Informationen bei der Validierung aktiviert. Der Standardwert ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- verbose-output

  --verbose-output = *FILE*
  Schreibt die ausführliche Ausgabe in *FILE*.

- version

  --version
  Zeigt die Version von RaptorXML Server an. Setzen Sie `--version` bei Verwendung mit einem Befehl vor den Befehl.

- warning-limit

  --warning-limit = *N | unlimited*
  Definiert das Warnungslimit im Bereich von `1-65535` oder `unbegrenzt`. Bei Erreichen dieses Limits wird die Verarbeitung fortgesetzt, doch werden keine weiteren Warnungen mehr ausgegeben. Der Standardwert ist `100`.

5.11.3 Verarbeitung

- listfile

  --listfile = `true|false`
  Bei `true` wird das Argument *InputFile* des Befehls als Textdatei behandelt, die einen Dateinamen pro Zeile enthält. Der Standardwert ist `false`. (Als Alternative können die Dateien im CLI getrennt durch ein Leerzeichen aufgelistet werden. Beachten Sie allerdings, dass CLIs eine maximale Zeichenanzahl haben). Beachten Sie, dass die Option `--listfile` nur auf Argumente, nicht aber auf Optionen angewendet wird.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- parallel-assessment [pa]

  --pa | --parallel-assessment = `true|false`
  Bei Setzung auf `true` wird die Schema-Validierung parallel ausgeführt. Das bedeutet, wenn sich auf irgendeiner Ebene mehr als 128 Elemente befinden, so werden diese Elemente über mehrere Threads parallel verarbeitet. Auf diese Weise können besonders große XML-Dateien schneller verarbeitet werden, wenn diese Option aktiv ist. Parallele Validierungen können gleichzeitig auf einer hierarchischen Ebene ausgeführt werden, können in einem einzigen Infoset aber auch auf mehreren Ebenen erfolgen. Beachten Sie dass die parallele Validierung im Streaming-Modus nicht funktioniert. Aus diesem Grund wird die Option `--streaming` ignoriert, wenn `--parallel-assessment` auf `true`. gesetzt ist. Außerdem wird bei Verwendung der Option `--parallel-assessment` mehr Arbeitsspeicher benötigt. Die Standardeinstellung ist `false`. Die Kurzform für die Option ist `--pa`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- script

  --script = *FILE*
  Führt nach Abschluss der Validierung das Python-Skript in der angegebenen Datei aus. Fügen Sie die

Option mehrmals hinzu, um mehr als ein Skript zu definieren.

▼ script-api-version

--api, --script-api-version = `1; 2; 2.1 to 2.4; 2.4.1; 2.5 bis 2.8; 2.8.1 bis 2.8.6; 2.9.0; 2.10.0; 2.11.0`
Definiert, welche Python API-Version für das Skript verwendet werden soll. Der Standardwert ist die neueste Version, derzeit **`2.11.0`**. Anstelle von Ganzzahlwerten wie `1` und `2` können Sie auch die entsprechenden Werte `1.0` und `2.0` verwenden. Ebenso können Sie anstelle der zwei Ziffern `2.5` die drei Ziffern `2.5.0` verwenden. Siehe auch Kapitel Python API-Versionen 392.

▼ script-param

--script-param = *KEY:VALUE*
Zusätzliche benutzerdefinierte Parameter, die während der Ausführung von Python Skripts aufgerufen werden können. Fügen Sie die Option mehrmals hinzu, um mehr als einen Parameter zu definieren.

▼ streaming

--streaming = `true|false`
Aktiviert die Streaming-Validierung. Standardwert ist `true`. Die im Arbeitsspeicher gehaltene Datenmenge wird im Streaming-Modus minimiert. Der Nachteil ist, dass später eventuell benötigte Informationen - z.B. ein Datenmodell des XML-Instanzdokuments - nicht mehr verfügbar sind. In Situationen, in denen dies eine Rolle spielt, muss der Streaming-Modus deaktiviert werden (indem Sie `--streaming` auf den Wert `false` setzen). Wenn Sie die Option `--script` mit dem Befehl `valxml-withxsd` verwenden, sollten Sie das Streaming deaktivieren. Beachten Sie, dass die Option `--streaming` ignoriert wird, wenn `--parallel-assessment` auf `true` gesetzt wird.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

▼ xml-validation-error-as-warning

--xml-validation-error-as-warning = `true|false`
Bei `true` werden Validierungsfehler als Warnungen behandelt. Wenn Fehler als Warnungen behandelt werden, wird die weitere Verarbeitung, z.B. eine XSLT-Transformation ungeachtet der Fehler fortgesetzt. Die Standardeinstellung ist `false`.

5.11.4 XML

▼ assessment-mode

--assessment-mode = `lax|strict`
Definiert den Beurteilungsmodus für die Gültigkeit von Schemas gemäß der XSD-Spezifikation. Der Standardwert ist `strict`. Das XML-Instanzdokument wird entsprechend dem mit dieser Option definierten Modus validiert.

▼ dtd

--dtd = *FILE*
Definiert das für die Validierung zu verwendende externe DTD-Dokument. Wenn das XML-Dokument eine

Referenz auf eine externe DTD enthält, setzt die CLI-Option die externe Referenz außer Kraft.

- load-xml-with-psvi

  --load-xml-with-psvi = `true|false`
  Ermöglicht die Validierung von XML-Input-Dateien und die Generierung von Informationen für diese Dateien nach Validierung des Schemas. Der Standardwert ist: `true`.

- namespaces

  --namespaces = `true|false`
  Aktiviert die Verarbeitung unter Berücksichtigung des Namespace. Dies ist nützlich, um die XML-Instanz auf Fehler aufgrund falscher Namespaces zu überprüfen. Der Standardwert ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- xinclude

  --xinclude = `true|false`
  Aktiviert die Unterstützung für XML-Inkludierungen (XInclude). Der Standardwert ist `false`. Bei `false` werden die `include`-Elemente von XInclude ignoriert.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- xml-mode

  --xml-mode = `wf|id|valid`
  Definiert, welcher XML-Verarbeitungsmodus für das XML-Instanzdokument verwendet werden soll: `wf`=Wohlgeformtheitsprüfung; `id`=Wohlgeformtheitsprüfung mit ID/IDREF-Prüfung; `valid`=Validierung. Der Standardwert ist `wf`. Beachten Sie, dass jedes bei der Verarbeitung geladene Instanzdokument eine DTD referenzieren muss, damit der Wert `valid` ausgegeben werden kann. Falls keine DTD vorhanden ist, wird ein Fehler ausgegeben.

- xml-validation-error-as-warning

  --xml-validation-error-as-warning = `true|false`
  Bei `true` werden Validierungsfehler als Warnungen behandelt. Wenn Fehler als Warnungen behandelt werden, wird die weitere Verarbeitung, z.B. eine XSLT-Transformation ungeachtet der Fehler fortgesetzt. Die Standardeinstellung ist `false`.

- xsd

  --xsd = *FILE*
  Definiert ein oder mehrere XML-Schema-Dokumente, die für die Validierung von XML-Instanzdokumenten verwendet werden sollen. Um mehr als ein Schema-Dokument zu definieren, fügen Sie die Option mehrmals hinzu.

5.11.5 XSD

assessment-mode

--assessment-mode = lax|strict
Definiert den Beurteilungsmodus für die Gültigkeit von Schemas gemäß der XSD-Spezifikation. Der Standardwert ist strict. Das XML-Instanzdokument wird entsprechend dem mit dieser Option definierten Modus validiert.

ct-restrict-mode

--ct-restrict-mode = 1.0|1.1|default
Definiert, wie complexType-Einschränkungen überprüft werden sollen. Beim Wert **1.0** werden complexType-Einschränkungen anhand der XSD 1.0-Spezifikation überprüft - und zwar auch im XSD 1.1-Validierungsmodus. Beim Wert **1.1** werden complexType-Einschränkungen anhand der XSD 1.1-Spezifikation überprüft - und zwar auch im XSD 1.0-Validierungsmodus. Beim Wert **default** werden complexType-Einschränkungen anhand der als aktueller Validierungsmodus (1.0 oder 1.1) ausgewählten XSD-Spezifikation überprüft. Der Standardwert ist **default**.

namespaces

--namespaces = true|false
Aktiviert die Verarbeitung unter Berücksichtigung des Namespace. Dies ist nützlich, um die XML-Instanz auf Fehler aufgrund falscher Namespaces zu überprüfen. Der Standardwert ist false.
Hinweis: Die Booleschen Optionswerte werden auf true gesetzt, wenn die Option ohne einen Wert definiert wird.

report-import-namespace-mismatch-as-warning

--report-import-namespace-mismatch-as-warning = true|false
Stuft Fehler, die beim Import von Schemas mit xs:import aufgrund eines nicht übereinstimmenden Namespace oder Ziel-Namespace auftreten, von Fehlern auf Warnungen herab. Der Standardwert ist false.
Hinweis: Die Booleschen Optionswerte werden auf true gesetzt, wenn die Option ohne einen Wert definiert wird.

schema-imports

--schema-imports = load-by-schemalocation | load-preferring-schemalocation | load-by-namespace | load-combining-both | license-namespace-only
Definiert das Verhalten von xs:import Elementen, von denen jedes ein optionales namespace Attribut und ein optionales schemaLocation Attribut hat: <import namespace="someNS" schemaLocation="someURL">. Mit der Option wird definiert, ob ein Schema-Dokument geladen oder nur ein Namespace lizenziert werden soll und, wenn ein Schema-Dokument geladen werden soll, anhand welcher Informationen es gesucht werden soll. Standardeinstellung: load-preferring-schemalocation. Das Verhalten ist das folgende:

- load-by-schemalocation: Anhand des Werts des schemaLocation Attributs wird der Schemapfad ermittelt, wobei Katalog-Mappings[48] berücksichtigt werden. Wenn das Namespace-Attribut vorhanden ist, wird der Namespace importiert (lizenziert).
- load-preferring-schemalocation: Wenn das schemaLocation Attribut vorhanden ist, wird es verwendet, wobei Katalog-Mappings[48] berücksichtigt werden. Falls kein schemaLocation Attribut vorhanden ist, wird der Wert des namespace Attributs über ein Katalog-Mapping[48] verwendet. Dies

ist der **Standardwert**.

- `load-by-namespace`: Anhand des Werts des `namespace` Attributs wird der Schemapfad über ein Katalog-Mapping[48] ermittelt.
- `load-combining-both`: Wenn entweder das Attribut `namespace` oder das Attribut `schemaLocation` ein Katalog-Mapping[48] hat, so wird das Mapping verwendet. Wenn beide Attribute Katalog-Mappings[48] haben, ist es vom Wert der Option `--schema-mapping` (XML/XSD-Option[255]) abhängig, welches Mapping verwendet wird. Falls kein Katalog-Mapping[48] vorhanden ist, wird das `schemaLocation` Attribut verwendet.
- `license-namespace-only`: Der Namespace wird importiert. Kein Schema-Dokument wird importiert.

▼ schema-location-hints

`--schema-location-hints =` `load-by-schemalocation | load-by-namespace | load-combining-both | ignore`

Definiert das Verhalten der Attribute `xsi:schemaLocation` und `xsi:noNamespaceSchemaLocation`: Legt fest, ob ein Schema-Dokument geladen werden soll und falls ja, anhand welcher Informationen es gesucht werden soll. Standardeinstellung: `load-by-schemalocation`.

- Der Wert `load-by-schemalocation` verwendet die URL des Schemapfads[422] in den Attributen `xsi:schemaLocation` und `xsi:noNamespaceSchemaLocation` in XML-Instanzdokumenten. Dies ist der **Standardwert**.
- Der Wert `load-by-namespace` verwendet den Namespace-Teil[422] von `xsi:schemaLocation` und - im Fall von `xsi:noNamespaceSchemaLocation` - einen leeren String und ermittelt das Schema über ein Katalog-Mapping[48].
- Bei Verwendung von `load-combining-both` und wenn entweder der Namespace-Teil oder der URL-Teil ein Katalog-Mapping[48] hat, so wird das Katalog-Mapping[48] verwendet. Wenn beide Katalog-Mappings[48] haben, ist es vom Wert der `--schema-mapping` Option (XML/XSD-Option[255]) abhängig, welches Mapping verwendet wird. Wenn weder der Namespace noch die URL ein Katalog-Mapping hat, wird die URL verwendet.
- Wenn der Wert der Option `ignore` ist, werden die beiden Attribute `xsi:schemaLocation` und `xsi:noNamespaceSchemaLocation` ignoriert.

▼ schema-mapping

`--schema-mapping =` `prefer-schemalocation | prefer-namespace`

Wenn ein Schema-Dokument sowohl anhand des Schema-Pfads als auch des Namespace gesucht werden soll, wird damit festgelegt, welche der beiden Optionen beim Katalog-Lookup Vorrang erhält. (Wenn eine der Optionen `--schemalocation-hints` oder `--schema-imports` einen Wert `load-combining-both` hat und wenn die betroffenen Namespace- und URL-Teile beide Katalog-Mappings[48] haben, gibt der Wert dieser Option an, welches der beiden Mappings verwendet werden soll (das Namespace Mapping oder das URL-Mapping; der Wert `prefer-schemalocation` bezieht sich auf das URL-Mapping)). Der Standardwert ist `prefer-schemalocation`.

▼ xml-mode-for-schemas

`--xml-mode-for-schemas =` `wf|id|valid`

Definiert, welcher XML-Verarbeitungsmodus für XML-Schema-Dokumente verwendet werden soll: `wf`=Wohlgeformtheitsprüfung; `id`=Wohlgeformtheitsprüfung mit ID/IDREF-Prüfung; `valid`=Validierung. Der Standardwert ist `wf`. Beachten Sie, dass jedes bei der Verarbeitung geladene Schema-Dokument eine DTD referenzieren muss, damit der Wert `valid` ausgegeben werden kann. Falls keine DTD vorhanden ist, wird ein Fehler ausgegeben.

▼ xsd-version

`--xsd-version =` `1.0|1.1|detect`

Definiert die zu verwendende W3C Schema Definition Language (XSD) Version. Der Standardwert ist `1.0`. Diese Option eignet sich auch, um herauszufinden, inwiefern ein 1.0-kompatibles Schema nicht mit Schemaversion 1.1 kompatibel ist. Die Option `detect` ist eine Altova-spezifische Funktionalität. Mit dieser Option kann die Version des XML-Schema-Dokuments (`1.0` oder `1.1`) durch Lesen des Werts des `vc:minVersion` Attributs des `<xs:schema>` Elements des Dokuments ermittelt werden. Wenn der Wert des `@vc:minVersion` Attributs `1.1` ist, wird das Schema als Version `1.1` erkannt. Bei jedem anderen Wert wird das Schema als `1.0` erkannt. Bei jedem anderen Wert oder bei Fehlen des `@vc:minVersion` Attributs wird das Schema als Version `1.0` gelesen.

5.11.6 XQuery

- indent-characters

  `--indent-characters =` *VALUE*
  Definiert den Zeichenstring, der als Einrückung verwendet werden soll.

- input

  `--input =` *FILE*
  Die URL der zu transformierenden XML-Datei.

- keep-formatting

  `--keep-formatting =` `true|false`
  Behält die Formatierung des Zieldokuments so gut wie möglich bei. Der Standardwert ist: `true`.

- omit-xml-declaration

  `--omit-xml-declaration =` `true|false`
  Serialisierungsoption, mit der angegeben wird, ob die XML-Deklaration in der Ausgabe enthalten sein soll oder nicht. Wenn der Wert `true` ist, enthält das Ausgabedokument keine XML-Deklaration. Wenn der Wert `false` ist, wird eine XML-Deklaration inkludiert. Der Standardwert ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- output, xsltoutput

  `output =` *FILE*, **`xsltoutput =`** *FILE*
  Die URL der primären Ausgabedatei. So ist z.B. im Fall der Ausgabe mehrerer HTML-Dateien die primäre Ausgabedatei der Pfad der Eintrittspunkt-HTML-Datei. Zusätzliche Ausgabedateien wie z.B. generierte Bilddateien werden als `xslt-additional-output-files` angegeben. Wenn keine `--output` oder `--xsltoutput` Option definiert ist, wird die Ausgabe in die Standardausgabe geschrieben.

- output-encoding

  `--output-encoding =` *VALUE*
  Der Wert des Kodierungsattributs im Ausgabedokument. Gültige Werte sind die Namen im IANA-Zeichensatz-Register. Der Standardwert ist `UTF-8`.

- output-indent

--output-indent = `true|false`
Wenn der Wert `true` ist, wird die Ausgabe entsprechend ihrer hierarchischen Struktur eingerückt. Bei `false` gibt es keine hierarchische Einrückung. Der Standardwert ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

▼ output-method

--output-method = `xml|html|xhtml|text`
Definiert das Ausgabeformat. Der Standardwert ist `xml`.

▼ param [p]

--p | --param = *KEY:VALUE*

▣ ***XQuery***

Definiert den Wert eines externen Parameters. Ein externer Parameter ist im XQuery-Dokument mit der `declare variable` Deklaration gefolgt von einem Variablennamen und anschließend dem Schlüsselwort `external`, gefolgt von einem Semikolon deklariert. Beispiel:

```
declare variable $foo as xs:string external;
```

Aufgrund des Schlüsselworts `external` wird `$foo` zu einem externen Parameter, dessen Wert zur Laufzeit von einer externen Quelle aus übergeben wird. Der externe Parameter erhält mit dem CLI-Befehl einen Wert. Beispiel:

```
--param=foo:'MyName'
```

In der obigen Beschreibungsanweisung ist *KEY* der Name des externen Parameters, *VALUE* der als XPath-Ausdruck angegebene Wert des externen Parameters. Im CLI verwendete Parameter müssen im XQuery-Dokument deklariert sein. Wenn mehrere externe Parameter als Werte an das CLI übergeben werden, muss jeder eine separate `--param` Option erhalten. Wenn der XPath-Ausdruck Leerzeichen enthält, muss er in doppelte Anführungszeichen gesetzt werden.

▣ ***XSLT***

Definiert einen globalen Stylesheet-Parameter. *KEY* ist der Parametername, *VALUE* der als XPath-Ausdruck angegebene Parameterwert. Im CLI verwendete Parameter müssen im Stylesheet deklariert sein. Wenn mehrere Parameter verwendet werden, muss vor jedem Parameter die `--param` Option verwendet werden. Wenn der XPath-Ausdruck Leerzeichen enthält - ob im XPath-Ausdruck selbst oder in einem String-Literal im Ausdruck - muss er in doppelte Anführungszeichen gesetzt werden. Beispiel:

```
raptorxml xslt --input=c:\Test.xml --output=c:\Output.xml --
param=date://node[1]/@att1 --p=title:'stringwithoutspace' --param=title:"'string
with spaces'" --p=amount:456 c:\Test.xslt
```

▼ updated-xml

--updated-xml = `discard|writeback|asmainresult`
Definiert, wie die aktualisierte XML-Datei behandelt werden soll.

- `discard`: Die Aktualisierung wird verworfen und nicht in eine Datei geschrieben. Weder die Input-Datei noch die Output-Datei wird aktualisiert. Beachten Sie, dass dies die Standardeinstellung ist.
- `writeback`: Schreibt die Aktualisierung zurück in die mit der Option `--input` definierte XML-Input-Datei.
- `asmainresult`: Schreibt die Aktualisierung in die mit der Option `--output` definierte XML-Output-

Datei. Wenn die **--output** Option nicht definiert wurde, wird die Aktualisierung in die Standardausgabe geschrieben. In beiden Fällen wird die XML-Input-Datei nicht geändert.

Der Standardwert ist `discard`.

- xpath-static-type-errors-as-warnings

  --xpath-static-type-errors-as-warnings = `true|false`
  Bei `true` werden alle im statischen XPath-Kontext gefundenen Typ-Fehler auf eine Warnung herabgestuft. Während die Ausführung bei einem Fehler fehlschlagen würde, könnte sie bei einer Warnung fortgesetzt werden. Die Standardeinstellung ist `false`.

- xquery-update-version

  --xquery-update-version = `1|1.0|3|3.0|`
  Definiert, ob der XQuery-Prozessor XQuery Update Facility 1.0 oder XQuery Update Facility 3.0 verwenden soll. Der Standardwert ist `3`.

- xquery-version

  --xquery-version = `1|1.0|3|3.0|3.1`
  Gibt an, ob der XQuery-Prozessor XQuery 1.0 oder XQuery 3.0 verwenden soll. Der Standardwert ist 3.1.

5.11.7 XSLT

- chartext-disable

  --chartext-disable = `true|false`
  Deaktiviert Diagrammerweiterungen. Der Standardwert ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- dotnetext-disable

  --dotnetext-disable = `true|false`
  Deaktiviert .NET-Erweiterungen. Der Standardwert ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- indent-characters

  --indent-characters = *VALUE*
  Definiert den Zeichenstring, der als Einrückung verwendet werden soll.

- input

  --input = *FILE*
  Die URL der zu transformierenden XML-Datei.

- javaext-barcode-location

--javaext-barcode-location = *FILE*
Definiert den Pfad zum Ordner, der die Barcode-Erweiterungsdatei `AltovaBarcodeExtension.jar` enthält. Der Pfad muss in einer der folgenden Formen angegeben werden:

- als Datei-URI, z.B.: `--javaext-barcode-location="file:///C:/Program Files/Altova/RaptorXMLServer2025/etc/jar/"`
- als Windows-Pfad mit maskierten umgekehrten Schrägstrichen, z.B.: `--javaext-barcode-location="C:\\Program Files\\Altova\\RaptorXMLServer2025\\etc\\jar\\"`

▼ javaext-disable

--javaext-disable = `true|false`
Deaktiviert Java-Erweiterungen. Der Standardwert ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

▼ output, xsltoutput

output = *FILE*, **xsltoutput =** *FILE*
Die URL der primären Ausgabedatei. So ist z.B. im Fall der Ausgabe mehrerer HTML-Dateien die primäre Ausgabedatei der Pfad der Eintrittspunkt-HTML-Datei. Zusätzliche Ausgabedateien wie z.B. generierte Bilddateien werden als `xslt-additional-output-files` angegeben. Wenn keine `--output` oder `--xsltoutput` Option definiert ist, wird die Ausgabe in die Standardausgabe geschrieben.

▼ param [p]

--p | --param = *KEY:VALUE*

▣ ***XQuery***

Definiert den Wert eines externen Parameters. Ein externer Parameter ist im XQuery-Dokument mit der `declare variable` Deklaration gefolgt von einem Variablennamen und anschließend dem Schlüsselwort `external`, gefolgt von einem Semikolon deklariert. Beispiel:

```
declare variable $foo as xs:string external;
```

Aufgrund des Schlüsselworts `external` wird `$foo` zu einem externen Parameter, dessen Wert zur Laufzeit von einer externen Quelle aus übergeben wird. Der externe Parameter erhält mit dem CLI-Befehl einen Wert. Beispiel:

```
--param=foo:'MyName'
```

In der obigen Beschreibungsanweisung ist *KEY* der Name des externen Parameters, *VALUE* der als XPath-Ausdruck angegebene Wert des externen Parameters. Im CLI verwendete Parameter müssen im XQuery-Dokument deklariert sein. Wenn mehrere externe Parameter als Werte an das CLI übergeben werden, muss jeder eine separate `--param` Option erhalten. Wenn der XPath-Ausdruck Leerzeichen enthält, muss er in doppelte Anführungszeichen gesetzt werden.

▣ ***XSLT***

Definiert einen globalen Stylesheet-Parameter. *KEY* ist der Parametername, *VALUE* der als XPath-Ausdruck angegebene Parameterwert. Im CLI verwendete Parameter müssen im Stylesheet deklariert sein. Wenn mehrere Parameter verwendet werden, muss vor jedem Parameter die `--param` Option verwendet werden. Wenn der XPath-Ausdruck Leerzeichen enthält - ob im XPath-Ausdruck selbst oder in einem String-Literal im Ausdruck - muss er in doppelte Anführungszeichen gesetzt werden. Beispiel:

```
raptorxml xslt --input=c:\Test.xml --output=c:\Output.xml --
param=date://node[1]/@att1 --p=title:'stringwithoutspace' --param=title:"'string
```



```
with spaces'" --p=amount:456 c:\Test.xslt
```

- streaming

  --streaming = `true|false`
  Aktiviert die Streaming-Validierung. Standardwert ist `true`. Die im Arbeitsspeicher gehaltene Datenmenge wird im Streaming-Modus minimiert. Der Nachteil ist, dass später eventuell benötigte Informationen - z.B. ein Datenmodell des XML-Instanzdokuments - nicht mehr verfügbar sind. In Situationen, in denen dies eine Rolle spielt, muss der Streaming-Modus deaktiviert werden (indem Sie `--streaming` auf den Wert `false` setzen). Wenn Sie die Option `--script` mit dem Befehl `valxml-withxsd` verwenden, sollten Sie das Streaming deaktivieren. Beachten Sie, dass die Option `--streaming` ignoriert wird, wenn `--parallel-assessment` auf `true` gesetzt wird.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- initial-template, template-entry-point

  --initial-template, --template-entry-point = *VALUE*
  Gibt den Namen einer benannten Vorlage im XSLT-Stylesheet an, das der Eintrittspunkt der Transformation ist.

- initial-mode, template-mode

  --initial-mode, --template-mode = *VALUE*
  Definiert den Vorlagenmodus für die Transformation.

- xpath-static-type-errors-as-warnings

  --xpath-static-type-errors-as-warnings = `true|false`
  Bei `true` werden alle im statischen XPath-Kontext gefundenen Typ-Fehler auf eine Warnung herabgestuft. Während die Ausführung bei einem Fehler fehlschlagen würde, könnte sie bei einer Warnung fortgesetzt werden. Die Standardeinstellung ist `false`.

- xslt-version

  --xslt-version = `1|1.0|2|2.0|3|3.0|3.1`
  Definiert, ob der XSLT-Prozessor XSLT 1.0, XSLT 2.0 oder XSLT 3.0 verwenden soll. Der Standardwert ist `3`.

5.11.8 JSON/Avro

- additional-schema

  --additional-schema = *FILE*
  Definiert URIs eines zusätzlichen Schema-Dokuments. Das zusätzliche Schema wird vom Hauptschema geladen und kann vom Hauptschema aus über die Eigenschaft **id** oder **$id** des zusätzlichen Schemas referenziert werden.

- disable-format-checks

  --disable-format-checks = `true|false`

Deaktiviert die durch das format-Attribut erzwungene semantische Validierung. Der Standardwert ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

▼ jsonc

`--jsonc =` `true|false`
Aktiviert die Unterstützung für Kommentare in JSON-Dokumenten. Der Standardwert ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

▼ json-lines

`--json-lines =` `true|false`
Aktiviert die Unterstützung für JSON Lines (d.h. einen JSON-Wert pro Zeile). Der Standardwert ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

5.11.9 XML-Signaturen

▼ absolute-reference-uri

`--absolute-reference-uri =` `true|false`
Definiert, ob die URI des signierten Dokuments als absolute (`true`) oder relative (`false`) URI gelesen werden soll. Der Standardwert ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

▼ certname, certificate-name

`--certname, --certificate-name =` *VALUE*
Der Name des zum Signieren verwendeten Zertifikats.

Windows
Dies ist der **Subject-**Name eines Zertifikats aus dem ausgewählten **`--certificate-store`** (Zertifikatspeicher).

eispiel zum Aufliste der Zertifikate (unter PowerShell)

```
% ls cert://CurrentUser/My
PSParentPath: Microsoft.PowerShell.Security\Certificate::CurrentUser\My
Thumbprint Subject
---------- -------
C9DF64BB0AAF5FA73474D78B7CCFFC37C95BFC6C CN=certificate1
... CN=...
```

eispiel: **`--certificate-name==certificate1`**

Linux/MacOS
--certname definiert den Dateinamen eines PEM-kodierten X.509v3-Zertifikats mit dem Private Key. Solche Dateien haben die Erweiterung **.pem**.

eispiel: **--certificate-name==/path/to/certificate1.pem**

▼ certstore, certificate-store

--certstore, --certificate-store = *VALUE*
Der Pfad, unter dem das mit **--certificate-name** definierte Zertifikat gespeichert ist.

Windows
Der Name eines Zertifikatspeichers unter **cert://CurrentUser**. Die verfügbaren Zertifikatspeicher können mit Hilfe von **% ls cert://CurrentUser/** (unter PowerShell) aufgelistet werden. Die Zertifikate würden anschließend folgendermaßen aufgelistet:

```
Name : TrustedPublisher
Name : ClientAuthIssuer
Name : Root
Name : UserDS
Name : CA
Name : ACRS
Name : REQUEST
Name : AuthRoot
Name : MSIEHistoryJournal
Name : TrustedPeople
Name : MyCertStore
Name : Local NonRemovable Certificates
Name : SmartCardRoot
Name : Trust
Name : Disallowed
```

eispiel: **--certificate-store==MyCertStore**

Linux/MacOS
Die Option **--certstore** wird derzeit nicht unterstützt.

▼ digest, digest-method

--digest, --digest-method = sha1|sha256|sha384|sha512|base64
Der zur Berechnung des Digest-Werts an der XML-Input-Datei angewendete Algorithmus. Verfügbare Werte sind: sha1|sha256|sha384|sha512.

▼ hmackey, hmac-secret-key

--hmackey, --hmac-secret-key = *VALUE*
Der Shared Secret HMAC-Schlüssel; muss mindestens sechs Zeichen lang sein.

Beispiel: **--hmackey=secretpassword**

▼ hmaclen, hmac-output-length

`--hmaclen, --hmac-output-length =` *LENGTH*
Kürzt die Ausgabe des HMAC-Algorithmus auf `length` Bits. Falls definiert, muss dieser Wert
- ein Vielfaches von 8 sein
- größer als 80 sein
- mehr als die Hälfte der Ausgabelänge des zugrunde liegenden Hash-Algorithmus betragen

▼ keyinfo, append-keyinfo

`--keyinfo, --append-keyinfo =` `true|false`
Definiert, ob das **`KeyInfo`**-Element in der Signatur inkludiert werden soll oder nicht. Der Standardwert ist **`false`**.

▼ sigc14nmeth, signature-canonicalization-method

`--sigc14nmeth, --signature-canonicalization-method =` *VALUE*
Definiert, welcher Kanonisierungsalgorithmus auf das Element **`SignedInfo`** angewendet werden soll. Der Wert muss einer der folgenden sein:

- `REC-xml-c14n-20010315`
- `xml-c14n11`
- `xml-exc-c14n#`

▼ sigmeth, signature-method

`--sigmeth, --signature-method =` *VALUE*
Definiert, welcher Algorithmus zum Generieren der Signatur verwendet werden soll.

Wenn ein Zertifikat verwendet wird
Wenn ein Zertifikat definiert ist, dann ist `SignatureMethod` optional und der Wert dieses Parameters wird vom Zertifikat abgeleitet. Wenn die Option definiert ist, muss sie mit dem vom Zertifikat verwendeten Algorithmus übereinstimmen.
eispiel: `rsa-sha256`.

Wenn --hmac-secret-key verwendet wird
Wenn `HMACSecretKey` verwendet wird, ist diese `SignatureMethod` obligatorisch. Der Wert muss einer der unterstützten HMAC-Algorithmen sein:
- `hmac-sha256`
- `hmac-sha386`
- `hmac-sha512`
- hmac-sha1 (soll laut Spezifikation nicht verwendet werden)

eispiel: **`hmac-sha256`**

▼ sigtype, signature-type

`--sigtype, --signature-type =` `detached | enveloping | enveloped`
Definiert den Typ der zu generierenden Signatur.

▼ transforms

--transforms = *VALUE*
Definiert, welche XML-Signaturtransformation auf das Dokument angewendet werden soll. Die unterstützten Werte sind die folgenden:

- `REC-xml-c14n-20010315` für Canonical XML 1.0 (Kommentare weglassen)
- `xml-c14n11` für Canonical XML 1.1 (Kommentare weglassen)
- `xml-exc-c14n#` für Exclusive XML Canonicalization 1.0 (Kommentare weglassen)
- `REC-xml-c14n-20010315#WithComments` für Canonical XML 1.0 (mit Kommentaren)
- `xml-c14n11#WithComments` for Canonical XML 1.1 (mit Kommentaren)
- `xml-exc-c14n#WithComments` for Exclusive XML Canonicalization 1.0 (mit Kommentaren)
- `base64`
- `strip-whitespaces` Altova-Erweiterung

eispiel: **--transforms=xml-c14n11**

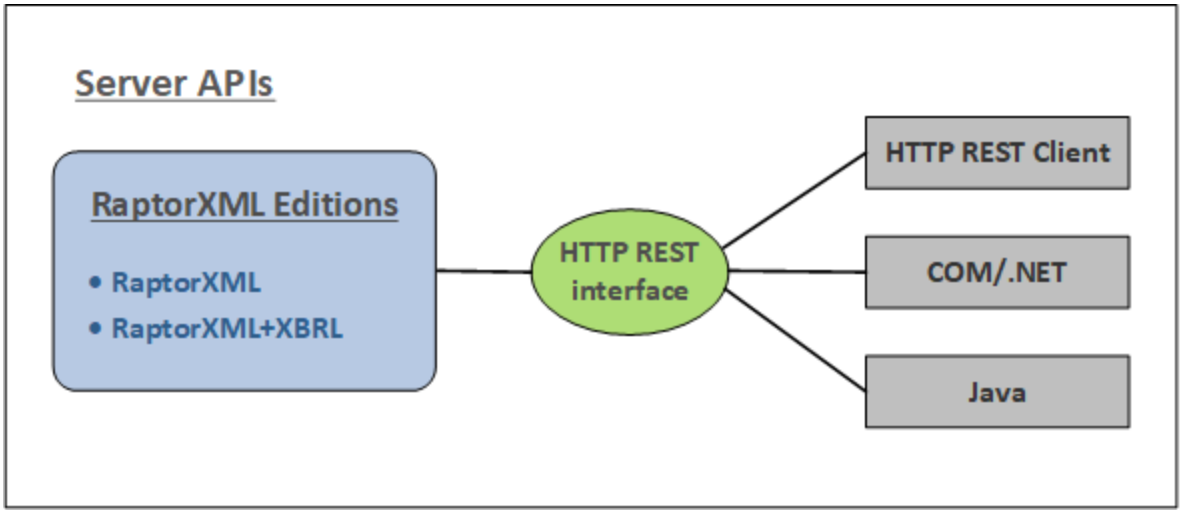
Anmerkung: Diese Option kann mehrmals definiert werden. Falls sie mehrmals definiert ist, spielt die Reihenfolge, in der sie definiert ist, eine Rolle. Die erste definierte Transformation gilt für das Input-Dokument. Die letzte definierte Transformation wird unmittelbar vor der Berechnung des Digest-Werts verwendet.

write-default-attributes

--write-default-attributes = `true|false`
Definiert, ob Standardattributwerte aus der DTD im signierten Dokument inkludiert werden sollen.

6 Server APIs; HTTP REST, COM/.NET, Java

In RaptorXML Server ist eine HTTP REST-Schnittstelle definiert, über die Clients Aufträge an den Server weitergeben können. Clients können die HTTP REST-Schnittstelle entweder direkt oder über die oberste Ebene der COM/.NET- und Java Server API aufrufen. Diese APIs bieten einfach zu verwendende COM/.NET- und Java-Klassen zur Verwaltung der Erstellung und Absendung der HTTP REST Requests.



Es gibt drei Server APIs, mit deren Hilfe über die HTTP REST-Schnittstelle mit RaptorXML kommuniziert werden kann (*siehe auch Abbildung oben*).

- HTTP REST-Client-Schnittstelle 268
- COM/.NET API 311
- Java API 320

Anmerkung: Die Server APIs bieten ähnliche Funktionalitäten wie die Befehlszeilenschnittstelle (CLI) 58. Dazu gehören die Validierung und Transformation von Dokumenten. Wenn Sie komplexere Funktionalitäten wie z.B. Auslesen, Extrahieren und Analysieren von Daten verwenden möchten, sollten Sie die Prozessor APIs verwenden. Über die Prozessor APIs stehen Ihnen zusätzliche Informationen wie z.B. die Anzahl der Elemente, deren Position im Dokument und der Zugriff auf und die Bearbeitung von komplexen XBRL-Daten zur Verfügung.

Verwendung

RaptorXML Server sollte auf einem Rechner installiert werden, auf den Clients über das lokale Netzwerk Zugriff haben. Nachdem der RaptorXML Server-Dienst gestartet wurde, können sich Clients mit dem Server verbinden und Befehle an diesen senden. Die folgenden Zugriffsmethoden werden als Server APIs bezeichnet, weil sie eine Methode bieten, mit einem entfernten RaptorXML Server zu kommunizieren.

- *HTTP REST-Client-Schnittstelle* 268: Die Client Requests erfolgen, wie im Abschnitt HTTP REST-Client-Schnittstelle 268 beschrieben, im JSON-Format. Jedem Request wird auf dem Server ein Auftragsverzeichnis, in dem Ausgabedateien gespeichert werden, zugewiesen. Der Server antwortet dem Client mit allen auftragsrelevanten Informationen.
- *COM/.NET API* 311 *und Java API* 320: Applikationen und Skripts in COM/.NET-Prgrammiersprachen 311 und Java 320-Applikationen verwenden Objekte der RaptorXML Server API 323, um Funktionalitäten von

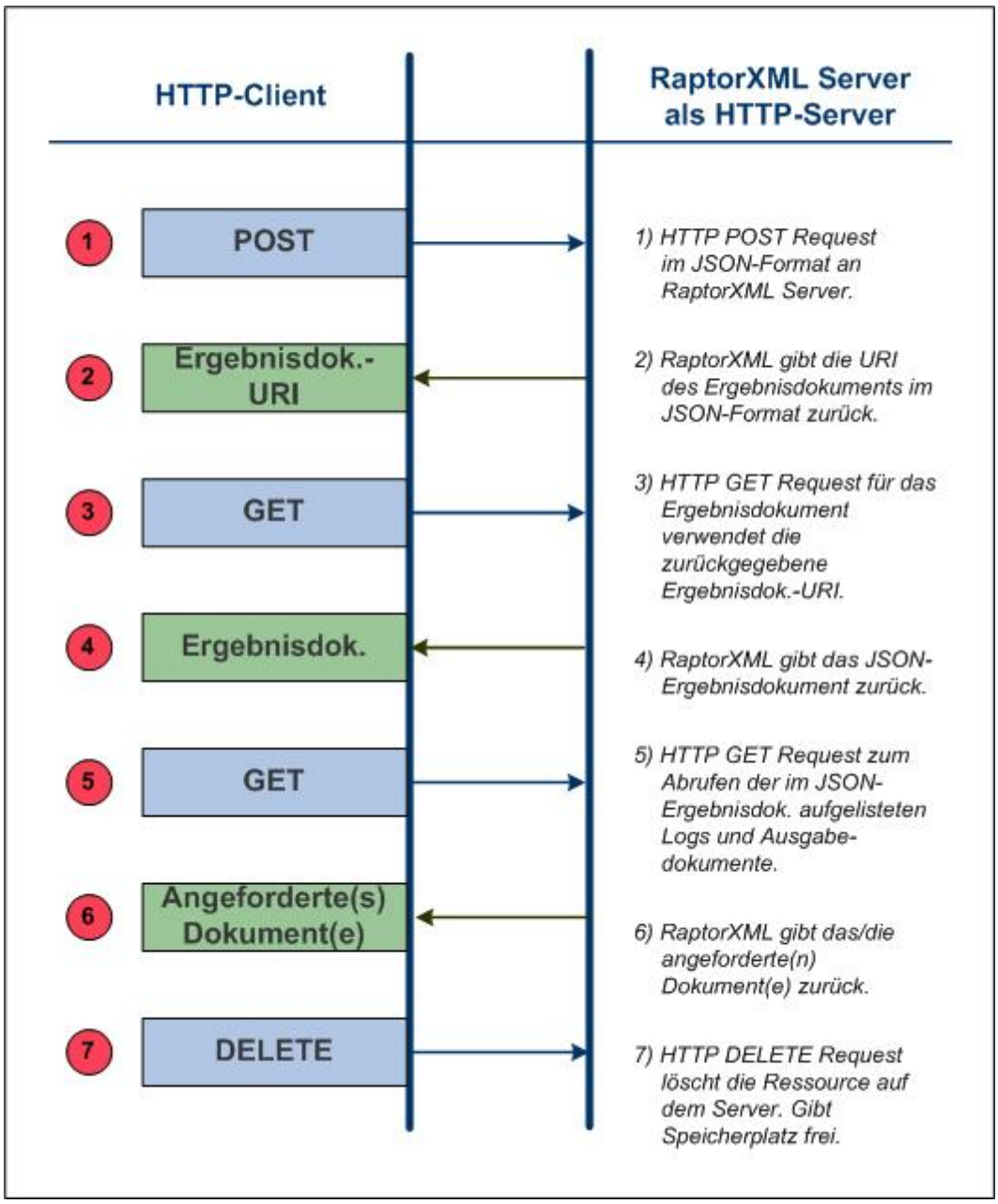
RaptorXML Server aufzurufen. Die RaptorXML Server API[323] erstellt die entsprechenden HTTP REST Requests für den Client. Nähere Informationen dazu finden Sie in den Unterabschnitten dazu.

Lizenzierung

RaptorXML Server wird auf dem Rechner, auf dem er installiert ist, lizenziert. Die Verbindungen zu RaptorXML Server erfolgen über HTTP.

6.1 HTTP REST-Client-Schnittstelle

RaptorXML Server nimmt über HTTP (oder HTTPS[278]) gesendete Aufträge an. Die Übertragung der Auftragsbeschreibung sowie der Ergebnis erfolgt im JSON-Format. Im unten gezeigten Diagramm sehen Sie eine Beschreibung des Ablaufs.



Sicherheitsfragen im Zusammenhang mit der HTTP REST-Schnittstelle

Standardmäßig können Ergebnisdokumente über die HTTP REST-Schnittstelle in jeden durch den Client angegebenen Ordner (auf den über das HTTP-Protokoll Zugriff besteht) geschrieben werden. Beim Konfigurieren von RaptorXML Server sollte dieser Sicherheitsaspekt daher berücksichtigt werden.

Falls die Sicherheit eventuell gefährdet sein könnte oder jemand über die Schnittstelle unbefugten Zugriff erhalten könnte, kann der Server so konfiguriert werden, dass Ergebnisdokumente in ein eigenes Ausgabeverzeichnis auf dem Server selbst geschrieben werden. Dies geschieht durch Setzen der Option `server.unrestricted-filesystem-access`[274] der Server-Konfigurationsdatei auf `false`. Wenn der Zugriff auf diese Weise eingeschränkt ist, kann der Client Ergebnisdokumente aus dem dafür vorgesehenen Ausgabeverzeichnis mit `GET` Requests herunterladen. Alternativ dazu kann ein Administrator die Ergebnisdokumentdateien vom Server in den Zielordner kopieren/laden.

In diesem Abschnitt

Bevor Sie einen Client Request senden, muss RaptorXML Server gestartet und ordnungsgemäß konfiguriert werden. Eine Beschreibung dazu finden Sie im Abschnitt Einrichten des Servers[269]. Eine Beschreibung dazu, wie man Client Requests sendet, finden Sie im Abschnitt Client Requests[282]. Im Abschnitt C#-Beispiel für die REST API[306] finden Sie eine Beschreibung der mit Ihrem RaptorXML Server-Paket installierten REST API-Beispieldatei.

6.1.1 Einrichten des Servers

RaptorXML muss auf dem Rechner, auf dem er installiert ist, lizenziert werden. Anschließend kann diese Installation über eine HTTP REST-Schnittstelle[268] aufgerufen werden. Um RaptorXML Server einzurichten, gehen Sie vor, wie im Folgenden beschrieben. Es wird davon ausgegangen, dass RaptorXML Server bereits korrekt installiert[47] und lizenziert[47] wurde.

1. Damit RaptorXML Server über HTTP oder HTTPS aufgerufen werden kann, muss das Produkt entweder als Dienst oder als Applikation gestartet sein[270]. Die Vorgehensweisen unterscheiden sich je nach Betriebssystem und sind hier für die folgenden Betriebssysteme beschrieben: Windows[270], Linux[271], macOS[271].
2. Verwenden Sie die Server-Anfangskonfiguration[272], um die Verbindung zum Server zu testen[271]. (Die Server-Anfangskonfiguration[272] ist die bei der Installation vordefinierte Standardkonfiguration.) Mit einem einfachen HTTP `GET` Request wie z.B. `http://localhost:8087/v1/version` können Sie die Verbindung testen. (Sie können den Request auch in die Adressleiste Ihres Browser-Fensters eingeben.) Wenn der Dienst ausgeführt wird, sollten Sie eine Antwort auf den HTTP Test-Request, wie z.B. den Version-Request oben, erhalten.
3. Öffnen Sie die Server-Konfigurationsdatei[272], `server_config.xml`. Wenn Sie Einstellungen[274] in der Datei ändern möchten, bearbeiten Sie die Server-Konfigurationsdatei und speichern Sie die

Änderungen. HTTPS ist standardmäßig deaktiviert und muss in der Konfigurationsdatei[272] aktiviert werden.

4. Falls Sie die Server-Konfigurationsdatei[272] bearbeitet haben, starten Sie den RaptorXML Server als Dienst neu, damit die neuen Konfigurationseinstellungen angewendet werden. Testen Sie anschließend die Verbindung erneut, um sicherzustellen, dass der Dienst ausgeführt wird und Sie ihn aufrufen können.

Anmerkung: Fehler, die beim Start des Servers auftreten, die verwendete Server-Konfigurationsdatei sowie Lizenzierungsfehler werden im System-Log protokolliert. Wenn Probleme mit dem Server auftreten, konsultieren Sie bitte das System-Log[274].

Nähere Informationen zu HTTPS finden Sie im Abschnitt HTTPS-Einstellungen[278].

6.1.1.1 Starten des Servers

In diesem Abschnitt werden folgende Schritte beschrieben:

- Pfad zur ausführbaren Server-Datei[270]
- Starten von RaptorXML als Dienst unter Windows[270]
- Starten von RaptorXML als Dienst unter Linux[271]
- Starten von RaptorXML als Dienst unter macOS[271]

Pfad zur ausführbaren Server-Datei

Die ausführbare RaptorXML Server-Datei ist standardmäßig im folgenden Ordner installiert:

```
<ProgramFilesFolder>\Altova\RaptorXMLServer2025\bin\RaptorXML.exe
```

Über die ausführbare Datei kann RaptorXML Server als Dienst gestartet werden.

Starten als Dienst unter Windows

Bei der Installation wurde RaptorXML Server als Dienst unter Windows registriert. Sie müssen RaptorXML Server allerdings als **Dienst starten**. Dazu gibt es die folgenden Möglichkeiten:

- über den Altova ServiceController, der in der Task-Leiste als Symbol zur Verfügung steht. Falls das Symbol nicht angezeigt wird, können Sie Altova ServiceController starten und sein Symbol zur Task-Leiste hinzufügen. Gehen Sie dazu zum **Startmenü** und wählen Sie **Alle Programme | Altova | Altova LicenseServer | Altova ServiceController**.
- über die Verwaltungskonsole für Windows-Dienste: **Systemsteuerung | Alle Systemsteuerungselemente | Verwaltung | Dienste**.
- über das Eingabeaufforderungsfenster, wenn es mit Administratorrechten geöffnet wurde. Verwenden Sie von jedem beliebigen Verzeichnis aus den folgenden Befehl: **`net start "AltovaRaptorXMLServer"`**
- über die ausführbare RaptorXML Server-Datei in einem Eingabeaufforderungsfenster: **`RaptorXMLServer.exe debug`**. Daraufhin wird der Server gestartet, wobei Informationen über die Serveraktivitäten direkt im Eingabeaufforderungsfenster angezeigt werden. Sie können die Informationen über die Serveraktivität über die Einstellung `http.log-screen`[274] der Server-Konfigurationsdatei[272] ein- und ausblenden. Drücken Sie **Strg+Untbr** (oder **Strg+Pause**), um den Server zu beenden. Wenn der Server anstatt als Dienst, wie im vorigen Schritt beschrieben, auf diese

Art gestartet wird, wird er beendet, wenn das Eingabeaufforderungsfenster geschlossen wird oder sich der Benutzer abmeldet.

Starten als Dienst unter Linux

Starten Sie RaptorXML Server mit dem folgenden Befehl als Dienst:

[< Debian 8]	sudo /etc/init.d/raptorxmlserver start
[≥ Debian 8]	sudo systemctl start raptorxmlserver
[< CentOS 7]	sudo initctl start raptorxmlserver
[≥ CentOS 7]	sudo systemctl start raptorxmlserver
[< Ubuntu 15]	sudo initctl start raptorxmlserver
[≥ Ubuntu 15]	sudo systemctl start raptorxmlserver
[RedHat]	sudo initctl start raptorxmlserver

Um RaptorXML Server zu beenden, verwenden Sie:

[< Debian 8]	sudo /etc/init.d/raptorxmlserver stop
[≥ Debian 8]	sudo systemctl stop raptorxmlserver
[< CentOS 7]	sudo initctl stop raptorxmlserver
[≥ CentOS 7]	sudo systemctl stop raptorxmlserver
[< Ubuntu 15]	sudo initctl stop raptorxmlserver
[≥ Ubuntu 15]	sudo systemctl stop raptorxmlserver
[RedHat]	sudo initctl stop raptorxmlserver

Starten als Dienst unter macOS

Starten Sie RaptorXML Server mit dem folgenden Befehl als Dienst:

```
sudo launchctl load /Library/LaunchDaemons/com.altova.RaptorXMLServer2025.plist
```

Um RaptorXML Server zu beenden, verwenden Sie:

```
sudo launchctl unload /Library/LaunchDaemons/com.altova.RaptorXMLServer2025.plist
```

6.1.1.2 Testen der Verbindung

In diesem Abschnitt werden folgende Schritte beschrieben:

- GET Request zum Testen der Verbindung 272
- Server Response und Empfangsbereitschaft für JSON-Datenstruktur 273

GET Request zum Testen der Verbindung

Nachdem RaptorXML Server gestartet wurde, testen Sie die Verbindung mit einem `GET` Request. (Sie können diesen Request auch in die Adressleiste eines Browser-Fensters eingeben.)

```
http://localhost:8087/v1/version
```

Anmerkung: Die Schnittstelle und die Port-Nummer von RaptorXML Server ist in der Server-Konfigurationsdatei, `server_config.xml`, definiert. Eine Beschreibung dazu finden Sie im nächsten Abschnitt Server-Konfiguration 272.

Server Response und Empfangsbereitschaft für JSON-Datenstruktur

Wenn der Dienst ausgeführt wird und der Server richtig konfiguriert ist, sollte der Request immer funktionieren. RaptorXML Server gibt seine Versionsinformationen als JSON-Datenstruktur zurück (*Codefragment unten*).

```
{
    "copyright": "Copyright (c) 1998-2013 Altova GmbH. ...",
    "name": "Altova RaptorXML+XBRL Server 2013 rel. 2 sp1",
    "eula": "http://www.altova.com/server_software_license_agreement.html"
}
```

Anmerkung: Wenn Sie die Server-Konfiguration durch Bearbeiten der Server-Konfigurationsdatei 272 ändern, sollten Sie die Verbindung erneut testen.

6.1.1.3 Konfigurieren des Servers

In diesem Abschnitt:

Server-Konfigurationsdatei: Anfangseinstellungen

RaptorXML Server wird mit Hilfe einer Konfigurationsdatei namens `server_config.xml` konfiguriert. Diese Datei befindet sich standardmäßig unter:

```
C:\Programme (x86)\Altova\RaptorXMLServer2025\etc\server_config.xml
```

In der Anfangskonfiguration für RaptorXML Server sind die folgenden Einstellungen definiert:

- die Port-Nummer `8087` als Port für den Server.
- dass der Server nur lokale Verbindungen (`localhost`) empfängt.
- dass der Server die Ausgabedatei in folgenden Ordner schreibt: `C:\ProgramData\Altova\RaptorXMLServer2025\Output\`.

Andere Standardeinstellungen sind unter Empfangsbereitschaft 273 von `server_config.xml` weiter unten beschrieben.

Server-Konfigurationsdatei: Ändern der Anfangseinstellungen, Zurücksetzen auf die Anfangseinstellungen

Wenn Sie die Anfangseinstellungen ändern möchten, müssen Sie die Server-Konfigurationsdatei, `server_config.xml` (siehe unten 273) ändern, diese speichern und anschließend RaptorXML Server als Dienst neu starten.

Wenn Sie die ursprüngliche Server-Konfigurationsdatei wiederherstellen möchten, (sodass wieder die Anfangseinstellungen für den Server konfiguriert sind), führen Sie den Befehl `createconfig` aus:

```
RaptorXMLServer.exe createconfig
```

Bei Ausführung dieses Befehls werden die Anfangseinstellungen wiederhergestellt und die Datei `server_config.xml` wird damit überschrieben. Der Befehl `createconfig :-)` ist nützlich, wenn Sie die Server-Konfiguration auf die Anfangseinstellungen zurücksetzen möchten.

Server-Konfigurationsdatei: Codefragment und Einstellungen

Unten sehen Sie den Inhalt der Server-Konfigurationsdatei, **`server_config.xml`**, mit den Anfangseinstellungen. Die Einstellungen in dieser Datei sind unterhalb des Codes erläutert.

`server_config.xml`

```
<config xmlns="http://www.altova.com/schemas/altova/raptorxml/config"
    xsi:schemaLocation="http://www.altova.com/schemas/altova/raptorxml/config
    http://www.altova.com/schemas/altova/raptorxml/config.xsd"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">

    <language>en</language>
    <server.unrestricted-filesystem-access>true</server.unrestricted-filesystem-access>
    <server.output-root-dir>C:
    \ProgramData\Altova\RaptorXMLServer2025\output\</server.output-root-dir>
    <server.script-root-dir>C:\Program
    Files\Altova\RaptorXMLServer2025\etc\scripts\</server.script-root-dir>
    <!--<server.default-script-api-version>2</server.default-script-api-version>-->
    <!--<server.catalog-file>catalog.xml</server.catalog-file>-->
    <!--<server.log-file>C:
    \ProgramData\Altova\RaptorXMLServer2025\Log\server.log</server.log-file>-->

    <http.enable>true</http.enable>
    <http.environment>production</http.environment>
    <http.socket-host>127.0.0.1</http.socket-host>
    <http.socket-port>8087</http.socket-port>
    <http.log-screen>true</http.log-screen>
    <http.access-file>C:\ProgramData\Altova\RaptorXMLServer2025\Log\access.log</http.access-
    file>
    <http.error-file>C:\ProgramData\Altova\RaptorXMLServer2025\Log\error.log</http.error-
    file>
```

```
    <https.enable>false</https.enable>
    <https.socket-host>127.0.0.1</https.socket-host>
    <https.socket-port>443</https.socket-port>
    <https.private-key>C:\Program
    Files\Altova\RaptorXMLServer2025\etc\cert\key.pem</https.private-key>
    <https.certificate>C:\Program
    Files\Altova\RaptorXMLServer2025\etc\cert\cert.pem</https.certificate>
    <!--<https.certificate-chain>/path/to/chain.pem</https.certificate-chain>-->

    <syslog.enabled>true</syslog.enabled>
    <syslog.protocol>BSD_UDP</syslog.protocol>
    <syslog.host>localhost</syslog.host>
    <syslog.port>514</syslog.port>

</config>
```

Einstellungen

Das Einstellen sind in die folgenden Bereiche unterteilt: (i) Allgemeine Einstellungen; (ii) HTTP; (iii) HTTPS; (iv) Syslog.

Allgemeine Server-Einstellungen

language

Definiert die Sprache von Server-Meldungen in einem optionalen `language` Element. Der Standardwert ist `en` (Englisch). Zulässige Werte sind `en|de|es|fr|ja` (Englisch, Deutsch, Spanisch, Französisch, Japanisch). Eine kurze Einführung, wie Sie RaptorXML lokalisieren, finden Sie unter Lokalisierungsbefehle 225.

server.unrestricted-filesystem-access

- Wenn die Option auf `true` (den Standardwert) gesetzt ist, werden die Ausgabedateien direkt in den vom Benutzer und in Python-Skripts angegebenen Ordner geschrieben (wobei eventuell Dateien desselben Namens überschrieben werden). Beachten Sie jedoch, dass Dateien bei Zugriff über HTTP über einen entfernten Rechner nicht über lokale Pfade aufgerufen werden können. Wenn RaptorXML Server daher auf einem entfernten Rechner ausgeführt wird, setzen Sie den Wert dieser Option auf **`false`**. Der Wert sollte nur dann auf **`true`** gesetzt werden, wenn Client und Server sich auf demselben Rechner befinden und die Ausgabedateien in ein Verzeichnis auf diesem Rechner geschrieben werden sollen.
- Wenn der Wert auf `false` gesetzt wird, werden die Dateien im Ausgabeverzeichnis 274 in das Verzeichnis für den Auftrag geschrieben und die URI dieser Dateien wird in das Ergebnisdokument 301 inkludiert. Wenn Sie den Wert auf `false` setzen, ist die Sicherheitsstufe etwas höher, da die Dateien nur in ein bestimmtes Verzeichnis auf dem Server geschrieben werden können. Die Auftragsausgabedateien können anschließend überprüft und in einen anderen Ordner kopiert werden.

server.output-root-dir

Das Verzeichnis, in dem die Ausgabedateien aller gesendeten Aufträge gespeichert werden.

server.script-root-dir

Das Verzeichnis, in dem vertrauenswürdige Python Skripts 391 gespeichert werden sollen. Die Option `script` funktioniert bei Verwendung über die HTTP-Schnittstelle nur dann, wenn Skripts aus diesem

vertrauenswürdigen Verzeichnis verwendet werden. Wird ein Python Skript aus einem anderen Verzeichnis angegeben, wird ein Fehler zurückgegeben. *Siehe 'Python-Skripts sicher machen'*[391].

server.default-script-api-version
Die zum Ausführen von Python Skripts standardmäßig verwendete Python-API-Version. Standardmäßig wird die neueste Version der Python API verwendet. Derzeit werden die Werte 1 und 2 unterstützt.

server.catalog-file
Die URL der zu verwendenden XML-Katalogdatei. Standardmäßig wird die Katalogdatei `RootCatalog.xml` aus dem Ordner `<ProgramFilesFolder>\Altova\RaptorXMLServer2025\etc` verwendet. Verwenden Sie die `server.catalog-file` Einstellung nur dann, wenn Sie die Standard-Katalogdatei ändern möchten.

server.log-file
Der Name und Pfad der Server Log-Datei. Die Ereignisse auf dem Server wie z.B. *Server gestartet/beendet* werden ständig im Ereignis-Log des Systems protokolliert und in einem Ansichtsprogramm für Systemereignisse wie z.B. Windows Event Viewer angezeigt. Zusätzlich zur Anzeige im Ansichtsprogramm können Log-Meldungen auch in die mit der Option `server.log-file` definierte Datei geschrieben werden. Die Server-Log-Datei enthält Informationen über alle Aktivitäten auf dem Server wie z.B. Fehler beim Server-Start, die verwendeten Konfigurationsdateien und Lizenzfehler.

http
http.enable
Ein Boolescher Wert zum Aktivieren oder Deaktivieren von HTTP: `true | false`. HTTP kann unabhängig von HTTPS aktiviert/deaktiviert werden. Sowohl HTTP als auch HTTPS können gleichzeitig aktiv sein.

http.environment
Die internen Umgebungen von raptorxml: `production| development`. Die Development-Umgebung ist mehr auf die Bedürfnisse von Entwicklern ausgerichtet und ermöglicht einfacheres Debuggen als über die Production-Umgebung.

http.socket-host
Die Schnittstelle, über die RaptorXML Server aufgerufen wird. Falls RaptorXML Server auch Verbindungen mit entfernten Rechnern gestatten soll, kommentieren Sie das Element ein und setzen Sie den Inhalt auf: `0.0.0.0`. `<http.socket-host>0.0.0.0</http.socket-host>`. Dadurch kann der Dienst an jede ansprechbare Schnittstelle des Server-Rechners angebunden werden. Stellen Sie in diesem Fall sicher, dass die Firewall-Einstellungen entsprechend konfiguriert sind. Eingehende Firewall Exceptions für Altova-Produkte müssen folgendermaßen registriert sein: Altova LicenseServer: Port `8088`; Altova RaptorXML Server: Port `8087`; Altova FlowForce Server: Port `8082`.

http.socket-port
Der Port, über den der Dienst aufgerufen wird. Der Port muss festgelegt und bekannt sein, damit HTTP Requests korrekt an den Dienst adressiert werden können.

http.log-screen
Wenn RaptorXML Server mit dem Befehl `RaptorXMLServer.exe debug`, (siehe Starten des Servers 270) gestartet wird, und wenn `http.log-screen` auf `true` gesetzt ist, so wird die Serveraktivität im Eingabeaufforderungsfenster angezeigt. Andernfalls wird die Serveraktivität nicht angezeigt. Zusätzlich zum Schreiben von Log-Dateien, wird der Log-Bildschirm angezeigt.

http.access-file
Der Name und Pfad der HTTP-Zugriffsdatei. Die Zugriffsdatei enthält Informationen über Aktivitäten im Zusammenhang mit dem Zugriff. Sie enthält Informationen, die beim Beheben von Verbindungsproblemen hilfreich sein können.

http.error-file
Der Name und Pfad der HTTP-Fehlerdatei. Die Fehlerdatei enthält Fehler im Zusammenhang mit dem Netzwerkverkehr von und zum Server. Bei Verbindungsproblemen kann diese Datei nützliche Informationen für deren Behebung enthalten.

http.max_request_body_size
Mit dieser Option wird die Maximalgröße des in RaptorXML Server zulässigen Request Body in Byte definiert. Der Standardwert ist 100 MB. Wenn die Größe eines Request Body den für diese Option definierten Wert übersteigt, so antwortet der Server mit `HTTP Error 413: Request entity too large`. Der Wert der Option muss größer oder gleich Null sein. Der Grenzwert kann durch Setzen von `http.max_request_body_size=0` deaktiviert werden.

https
https.enable
Ein Boolescher Wert zum Aktivieren oder Deaktivieren von HTTPS: `true | false`. HTTPS kann unabhängig von HTTP aktiviert/deaktiviert werden. Sowohl HTTP als auch HTTPS können gleichzeitig aktiv sein. HTTPS ist standardmäßig deaktiviert und muss durch Änderung des Werts dieser Einstellung in **`true`** aktiviert werden.

https.socket-host
Erhält einen String-Wert, bei dem es sich um die Host-Adresse, unter der HTTPS-Verbindungen gestattet werden, handelt. Um nur Verbindungen vom lokalen Host-Rechner zu gestatten, definieren Sie **`localhost`** oder **`127.0.0.1`**. Wenn RaptorXML Server Verbindungen von allen entfernten Rechnern zulassen soll, definieren Sie als Wert `0.0.0.0` und zwar in folgender Form: `<https.socket-host>0.0.0.0</https.socket-host>`. Dadurch kann der Dienst an jede ansprechbare Schnittstelle des Server-Rechners angebunden werden. Stellen Sie in diesem Fall sicher, dass die Firewall-Einstellungen entsprechend konfiguriert sind. Eingehende Firewall Exceptions für Altova-Produkte müssen folgendermaßen registriert sein: Altova LicenseServer: Port `8088`; Altova RaptorXML Server: Port `8087`; Altova FlowForce Server: Port `8082`. Auch IPv6-Adressen wie `'::'` sind zulässig.

https.socket-port
Ein Ganzzahlwert, der den Port, auf dem HTTPS gestattet ist, angibt. Der Port muss festgelegt und bekannt sein, damit HTTP Requests korrekt an den Dienst adressiert werden können.

https.private-key, https.certificate
URIs, die die Pfade zum Private Key bzw. zu den Zertifikatdateien des Servers angeben. Beide URIs sind zwingend erforderlich. Nähere Informationen dazu finden Sie unter HTTPS-Einstellungen[278] und Einrichten der SSL-Verschlüsselung[279]. Auf Windows-Rechnern können auch Windows-Pfade verwendet werden.

https.certificate-chain
Dies ist eine optionale Einstellung, mit der der Pfad zu Zwischenzertifikatdateien angegeben wird. Wenn Sie zwei Zwischenzertifikate (ein primäres und ein sekundäres) haben, so kombinieren Sie diese zu einer Datei, wie in Schritt 7 unter Einrichten der SSL-Verschlüsselung[279] beschrieben. Nähere Informationen dazu finden Sie unter HTTPS-Einstellungen[278] und Einrichten der SSL-Verschlüsselung[279].

Syslog
syslog.enabled
Ein Boolescher Wert zum Aktivieren oder Deaktivieren des System-Logs: `true | false`. Der Standardwert ist `true`. Wenn der Server mit dem Debug-Befehl gestartet wird, wird diese Einstellung ignoriert und die Logs werden in der Konsole angezeigt.

syslog.protocol
Das für das Remote System Logging verwendete Protokoll: `BSD_UDP` oder `BSD_TCP`. Die Einstellung wird ignoriert, wenn **syslog.host** `localhost` (oder `127.0.0.1` oder `::1`) ist.

syslog.host
Der Name oder die IP-Adresse des Logging Host. Die Standardeinstellung ist `localhost`. Bei der Protokollierung auf `localhost` wird auf Windows-Systemen der Windows Event Logger verwendet. Für die Protokollierung auf `localhost` auf anderen Systemen wird Syslog (RFC3164) verwendet.

syslog.port
Ein Ganzzahlwert, der den Port angibt, auf dem der Syslog-Dienst Verbindungen gestattet. Der Port ist normalerweise `514` oder `601` oder `6514`. Der Standardwert ist `514`. Die Einstellung wird ignoriert, wenn **syslog.host** `localhost` (oder `127.0.0.1` oder `::1`) ist. Für die Protokollierung auf `localhost` wird auf Windows-Systemen der Windows Event Logger verwendet. Für die Protokollierung auf `localhost` auf anderen Systemen wird eine lokale Unix Socket-Verbindung verwendet.

Die RaptorXML Server-Adresse

Die HTTP-Adresse des Servers besteht aus dem Socket-Host und dem Socket-Port:

```
http://{socket-host}:{socket-port}/
```

In der Anfangskonfiguration lautet die Adresse folgendermaßen:

```
http://localhost:8087/
```

Um die Adresse zu ändern, ändern Sie in der Server-Konfigurationsdatei `server_config.xml` die Einstellungen von `http.socket-host` und `http.socket-port`. Angenommen, der Server-Rechner hat

die IP-Adresse `123.12.123.1` und es wurden die folgenden Sever-Konfigurationseinstellungen vorgenommen:

```
<http.socket-host>0.0.0.0</http.socket-host>
<http.socket-port>8087</http.socket-port>
```

RaptorXML Server kann in diesem Fall folgendermaßen adressiert werden:

```
http://123.12.123.1:8087/
```

Anmerkung: Nach Änderung von `server_config.xml` muss RaptorXML Server neu gestartet werden, damit die neuen Werte angewendet werden.

Anmerkung: Bei Problemen mit der Verbindung zu RaptorXML Server können die Informationen in den in `http.access-file` und `http.error-file` definierten Dateien beim Beheben der Probleme helfen.

Anmerkung: Nachrichten, die an RaptorXML Server gesendet werden, müssen Pfadnamen enthalten, die auf dem Server-Rechner gültig sind. Dokumente auf dem Server-Rechner können entweder lokal oder entfernt (z.B. über HTTP URIs) aufgerufen werden.

6.1.1.4 HTTPS-Einstellungen

RaptorXML Server kann nicht nur als HTTP-Server, sondern auch als HTTPS-Server gestartet werden. Beide Verbindungsarten können gleichzeitig aktiv sein.

Aktivieren von HTTPS

Standardmäßig ist die HTTPS-Unterstützung deaktiviert. Um HTTPS zu aktivieren, ändern Sie die Einstellung **`https.enable`** in der Serverkonfigurationsdatei[272], `server_config.xml`, in `true`. Ändern Sie die verschiedenen HTTPS-Einstellungen in der Konfigurationsdatei[272] entsprechend Ihren Server-Anforderungen.

Private Key und Zertifikat

Private Key und Zertifikatdateien enthalten Sie auf eine der folgenden Arten:

- Von einer Zertifizierungsstelle: Befolgen Sie die unter Einrichten der SSL-Verschlüsselung[279] beschriebene Anleitung.
- Erstellen Sie mit Hilfe des folgenden OpenSSL-Befehls (den Sie an Ihre Umgebung anpassen) ein eigensigniertes Zertifikat:

```
openssl req -x509 -newkey rsa:4096 -nodes -keyout key.pem -out cert.pem -days 365 -
subj "/C=AT/ST=vienna/L=vienna/O=Altova Gmbh/OU=dev/CN=www.altova.com"
```

Testen der Verbindung

Eine gute Methode zum Testen Ihrer Verbindung ist mit Hilfe des curl-Befehlszeilentools zum Übertragen von Daten über URLs. Sie können den folgenden Befehl verwenden:

```
curl.exe https://localhost:443/v1/version
```

Wenn das Zertifikat nicht vertrauenswürdig ist, verwenden Sie die Option `-k`, wie unten gezeigt:

```
curl.exe -k https://localhost:443/v1/version
```

Mit dem folgenden Befehl wird das mit RaptorXML Server bereitgestellte HTTP Python-Beispiel ausgeführt:

```
python3.exe examples\ServerAPI\python\RunRaptorXML.py --host localhost -p 443 -s
```

6.1.1.5 Einrichten der SSL-Verschlüsselung

Um Ihre RaptorXML Server-Datenübertragungen über das SSL-Protokoll zu verschlüsseln, sind folgende Schritte erforderlich:

- Generieren eines SSL Privat Key und Erstellen einer SS Public Key-Zertifikatdatei
- Konfigurieren von RaptorXML Server für die SSL-Kommunikation

Im Folgenden sind die Schritte im Einzelnen beschrieben.

Zur Verwaltung der SSL-Verschlüsselung wird der Open Source OpenSSL Toolkit verwendet. Die unten aufgelisteten Schritte müssen daher auf einem Computer durchgeführt werden, auf dem OpenSSL zur Verfügung steht. OpenSSL ist im Allgemeinen auf dem meisten Linux-Distributions und macOS-Rechnern vorinstalliert. Es kann auch auf Windows-Computern installiert werden. Download Links zu den Installations-Binärdateien finden Sie im OpenSSL Wiki.

Um einen Private Key zu generieren und ein Zertifikat von einer Zertifizierungsstelle zu erhalten, gehen Sie folgendermaßen vor:

1. Generieren eines Private Key

   Für SSL muss auf RaptorXML Server ein **Private Key** installiert werden. Mit Hilfe dieses Private Key werden alle an RaptorXML Server gesendeten Daten verschlüsselt. Verwenden Sie zum Erstellen des Private Key den folgenden OpenSSL-Befehl:

   ```
   openssl genrsa -out private.key 2048
   ```

   Dadurch wird eine Datei namens `private.key` generiert, die Ihren Private Key enthält. Merken Sie sich, wo Sie die Datei speichern. Anhand des Private Key wird der (i) Certificate Signing Request (CSR) generiert und der Private Key (ii) wird auf RaptorXML Server installiert (*siehe Schritt 8 weiter unten*).

2. Certificate Signing Requests (CSRs)

   Ein Certificate Signing Request (CSR) wird an eine Zertifizierungsstelle (Certificate Authority = CA) wie z.B. VeriSign oder Thawte gesendet, um ein Public Key-Zertifikat anzufordern. Der CSR basiert auf Ihrem Private Key und enthält Informationen über Ihr Unternehmen. Mit dem folgenden OpenSSL-Befehl (der die in Schritt 1 erstellte Private Key-Datei `private.key` als einen seiner Parameter enthält) wird ein CSR erstellt:

```
openssl req -new -nodes -key private.key -out my.csr
```

Während der Generierung des CSR müssen Sie die unten angeführten Informationen über Ihr Unternehmen angeben. Anhand dieser Informationen überprüft die Zertifizierungsstelle die Identität Ihres Unternehmens.

- *Country (Land)*
- *Locality (Ort)* (die Stadt, in dem Ihr Unternehmen seinen Firmensitz hat)
- *Organization (Unternehmen)* (Ihr Firmenname). Verwenden Sie keine Sonderzeichen, da sonst kein gültiges Zertifikat erstellt werden kann
- *Common Name* (der DNS-Name Ihres Servers). Dieser Name muss mit dem offiziellen Namen Ihres Servers, d.h. dem DNS-Namen, über den Client Apps eine Verbindung zum Server herstellen, genau übereinstimmen.
- *Ein "Challenge Password".* Dieser Eintrag muss leer bleiben!

3. Erwerben eines SSL-Zertifikats

   Erwerben Sie von einer anerkannten Zertifizierungsstelle (CA), wie z.B. VeriSign oder Thawte ein SSL-Zertifikat. In der restlichen Anleitung gehen wir nach dem VeriSign-Verfahren vor. Bei anderen CAs ist der Ablauf ähnlich.

   - Gehen Sie zur VeriSign Website.
   - Klicken Sie auf **Buy SSL Certificates**.
   - Es stehen unterschiedliche Arten von SSL-Zertifikaten zur Verfügung. Für RaptorXML Server genügen Secure Site- oder Secure Site Pro-Zertifikate. Eine EV (Extended Verification) ist nicht nötig, da Benutzern keine "grüne Adressleiste" angezeigt wird.
   - Erledigen Sie die Anmeldung und füllen Sie die erforderlichen Informationen für Ihre Bestellung aus.
   - Wenn Sie nach dem (*in Schritt 2 erstellten)* CSR gefragt werden, kopieren Sie den Inhalt der Datei `my.csr` in das Bestellformular.
   - Bezahlen Sie das Zertifikat mit Ihrer Kreditkarte.

   > **Wartezeit für das Zertifikat**
   >
   > Berücksichtigen Sie beim Einrichten von RaptorXML Server, dass es normalerweise **zwei bis drei Werktage** dauert, bis Sie Public Key-Zertifikate von einer SSL-Zertifizierungsstelle (CA) erhalten.

4. Zusendung des Public Key von der CA

   Die Zertifizierungsstelle benötigt zwei bis drei Werktage für die Bearbeitung Ihrer Bestellung. Während dieser Zeit erhalten Sie eventuell E-Mails oder Telefonanrufe, in denen überprüft wird, ob Sie berechtigt sind, ein SSL-Zertifikat für Ihre DNS-Domain zu erhalten. Beantworten Sie bitte die Fragen der CA, um das Zertifikat zu erhalten.

   Nach Abschluss des Überprüfungsverfahrens erhalten Sie eine E-Mail mit dem **Public Key** Ihres SSL-Zertifikats. Der Public Key wird entweder in Textform oder im Anhang in einer `.cer`-Datei gesendet.

5. Speichern des Public Key in einer Datei

   Um den Public Key mit RaptorXML Server verwenden zu können, muss er in einer **.cer**-Datei gespeichert werden. Wenn der Public Key in Textform gesendet wurde, kopieren Sie bitte alle Zeilen ab

   ```
   --BEGIN CERTIFICATE--
     ...
   --END CERTIFICATE--
   ```

   in eine Textdatei, z.B. in **mycertificate.cer**.

6. Speichern der Zwischenzertifikats der CA in einer Datei

   Zur Fertigstellung Ihres SSL-Zertifikats benötigen Sie zwei zusätzliche Zertifikate: das **primäre** und das **sekundäre Zwischenzertifikat**. Ihre Zertifizierungsstelle (CA) listet den Inhalt von Zwischenzertifikaten auf ihrer Website auf.

   - Die Zwischenzertifikate von Verisign: https://knowledge.verisign.com/support/ssl-certificates-support/index?page=content&id=AR657&actp=LIST&viewlocale=en_US
   - Die Zwischenzertifikate von Verisign für sein Secure Site-Produkt: https://knowledge.verisign.com/support/ssl-certificates-support/index?page=content&id=AR1735

   Kopieren Sie die beiden Zwischenzertifikate (das primäre und das sekundäre) in separate Textdateien und speichern Sie diese auf Ihrem Rechner.

7. Optionales Kombinieren der Zertifikate zu einer Public Key-Zertifikatdatei

   Sie haben nun drei Zertifikatdateien:

   - Public Key (`mycertificate.cer`)
   - Sekundäres Zwischenzertifikat
   - Primäres Zwischenzertifikat

   Sie können Ihre Zwischenzertifikate in Ihr Public Key-Zertifikat integrieren, wenn Sie möchten. Eine Anleitung dazu finden Sie unten. (Sie können den Pfad zu den Zwischenzertifikaten alternativ dazu auch mit Hilfe der **https.certificate-chain** Konfigurationsdateieinstellung[274] definieren.)

   Jede enthält Textblöcke innerhalb der folgenden Zeilen:

   ```
   --BEGIN CERTIFICATE--
     ...
   --END CERTIFICATE--
   ```

   Kopieren Sie nun alle drei Zertifikate der Reihe nach in eine Datei. Die richtige Reihenfolge ist wichtig: (i) Public Key, (ii) sekundäres Zwischenzertifikat, (iii) primäres Zwischenzertifikat. Stellen Sie sicher, dass sich keine Zeilen zwischen den Zertifikaten befinden.

   ```
   --BEGIN CERTIFICATE--
     Public Key aus mycertificate.cer (siehe Schritt 5)
   ```

```
--END CERTIFICATE--
--BEGIN CERTIFICATE--
  Sekundäres Zwischenzertifikat (siehe Schritt 6)
--END CERTIFICATE--
--BEGIN CERTIFICATE--
  Primäres Zwischenzertifikat (siehe Schritt 6)
--END CERTIFICATE--
```

Speichern Sie den kombinierten Zertifikattext in einer Datei namens **`publickey.cer`** . Dies ist die *Public Key-Zertifikatdatei* Ihres SSL-Zertifikats. Sie enthält Ihr Public Key-Zertifikat sowie die gesamte Kette der Zwischenzertifikate, mit denen die CA Ihr Zertifikat signiert hat. Die Public Key-Zertifikatdatei wird zusammen mit dem Private Key (*siehe Schritt 8)* auf RaptorXML Server installiert.

6.1.2 Client Requests

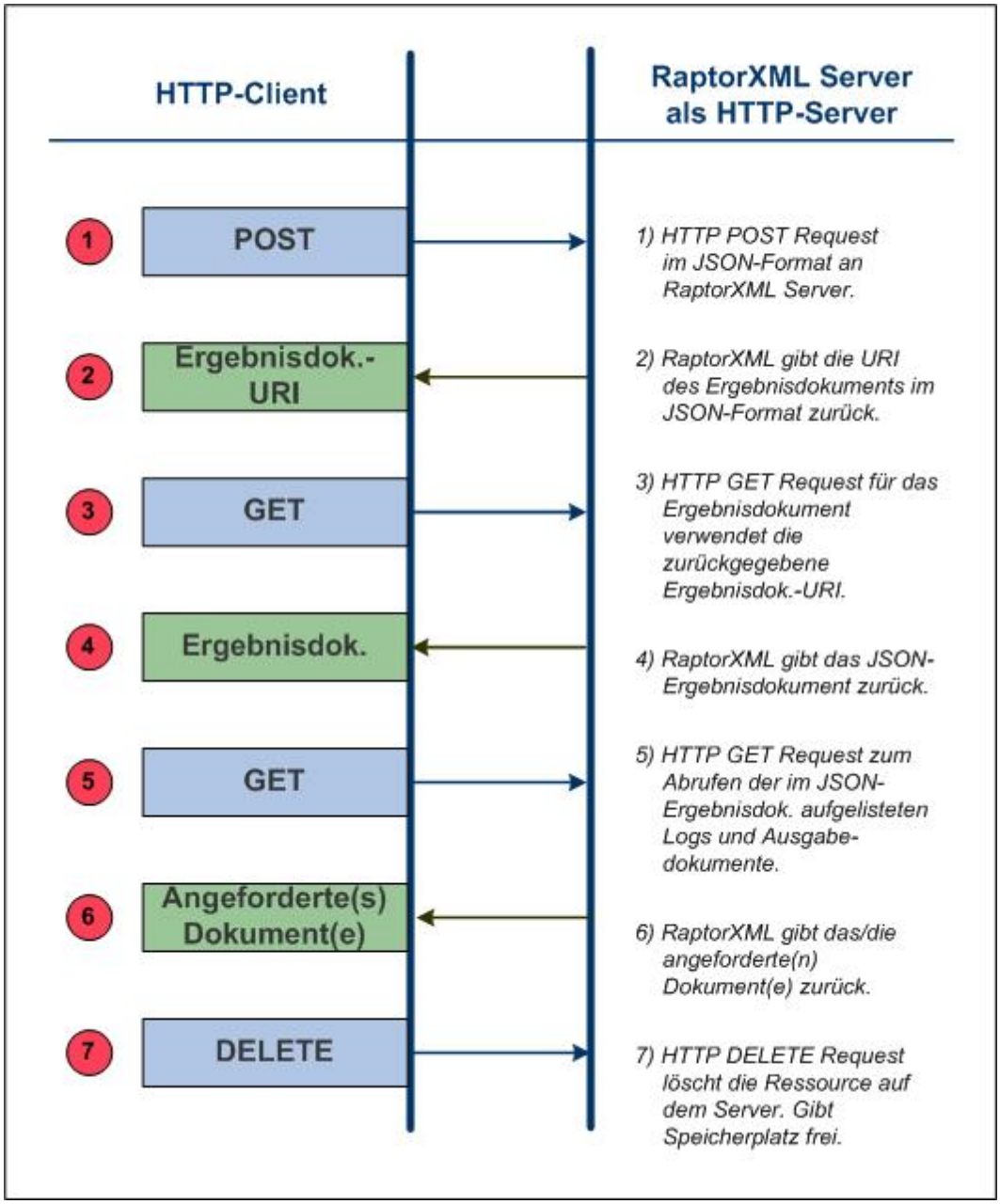
Nachdem RaptorXML Server als Dienst gestartet wurde 270, stehen seine Funktionalitäten jedem HTTP-Client zur Verfügung. Der HTTP-Client kann:

- die HTTP-Methoden `GET`, `PUT`, `POST` und `DELETE` verwenden
- das `Content-Type` Header-Feld definieren

Benutzerfreundlicher HTTP-Client

Im Internet steht ein ganze Reihe von Web Clients zum Download zur Verfügung. Wir haben RESTClient von Mozilla, einen benutzerfreundlichen und zuverlässigen Web Client, verwendet. Dieser Client kann als Plug-in zu Firefox hinzugefügt werden, ist einfach zu installieren, unterstützt die von RaptorXML benötigen HTTP-Methoden und bietet eine ausreichend gute JSON-Syntaxfärbung. Wenn Sie bisher noch nicht mit HTTP-Clients gearbeitet haben, empfehlen wir Ihnen RESTClient. Beachten Sie bitte, dass Sie RESTClient auf eigenes Risiko installieren und verwenden.

Ein typischer Client Request besteht, wie im Diagramm unten gezeigt, aus einer Reihe von Schritten.



Im Folgenden finden Sie einige wichtige Anmerkungen zu den einzelnen Schritten. Schlüsselbegriffe sind fett gedruckt.

1. Mit Hilfe einer HTTP `POST`-Methode wird ein Request[284] im **JSON-Format** erstellt. Der Request könnte für jede Funktionalität von RaptorXML Server sein. So könnte z.B. eine Validierung oder eine XSLT-Transformation angefordert werden. Die im Request verwendeten Befehle, Argumente und Optionen sind dieselben, die auch in der Befehlszeile[58] verwendet werden. Der Request wird auf: **`http://localhost:8087/v1/queue`** mittels `POST` bereitgestellt, wobei **`localhost:8087`** hier die Adresse von RaptorXML Server (die in der Anfangskonfiguration verwendete Server-Adresse[277]) ist. Ein solcher Request wird als **RaptorXML Server-Auftrag** bezeichnet.

2. Wenn der Request von RaptorXML Server erhalten und für die Verarbeitung akzeptiert wurde, wird nach Verarbeitung des Auftrags ein **Ergebnisdokument** mit den Ergebnissen der Server-Aktion erstellt. Die **URI dieses Ergebnisdokuments** (im Diagramm oben die Ergebnisdok.-URI) wird an den Client zurückgegeben. 301 Beachten Sie, dass die URI unmittelbar nach Übernahme des Auftrags für die Verarbeitung (nachdem er in die Warteschlange gestellt wurde) und auch, wenn die Verarbeitung noch nicht abgeschlossen wurde, zurückgegeben wird.

3. Der Client sendet (über die Ergebnisdokument-URI) in einer `GET`-Methode einen Request für das Ergebnisdokument 301 an den Server. Wenn der Auftrag zum Zeitpunkt des Empfangs des Request noch nicht gestartet oder noch nicht abgeschlossen wurde, gibt der Server den Status *Running* zurück. Der `GET` Request muss so oft wiederholt werden, bis der Auftrag fertig gestellt ist und das Ergebnisdokument erstellt wurde.

4. RaptorXML Server gibt das Ergebnisdokument im JSON-Format zurück 301. Das Ergebnisdokument kann die **URIs von Fehler- oder Ausgabedokumenten,** die von RaptorXML Server beim Verarbeiten des ursprünglichen Request erzeugt wurden, enthalten. So werden z.B. Fehlerprotokolle zurückgegeben, wenn bei einer Validierung Fehler ausgegeben wurden. Die primären Ausgabedokumente, wie z.B. das Ergebnis einer XSLT-Transformation werden zurückgegeben, wenn der Auftrag zur Erzeugung einer Ausgabe erfolgreich ausgeführt wurde.

5. Der Client sendet die URIs der in Schritt 4 erhaltenen Ausgabedokumente 304 über eine HTTP `GET`-Methode an den Server. Jeder Request wird in einer separaten `GET`-Methode gesendet.

6. RaptorXML Server gibt die angeforderten Dokumente 304 in Antwort auf die in Schritt 5 gesendeten `GET`-Requests zurück.

7. Der Client kann nicht benötigte Dokumente, die als Ergebnis eines Auftrags-Request auf dem Server generiert wurden, löschen 306. Zu diesem Zweck sendet er die URI des entsprechenden Ergebnisdokuments in einer HTTP `DELETE`-Methode. Daraufhin werden alle im Zusammenhang mit diesem Auftrag generierten Dateien von der Festplatte gelöscht. Dazu gehören das Ergebnisdokument, alle temporären Dateien sowie alle Fehler- und Ausgabedokumentdateien. Dadurch schaffen Sie Platz auf der Festplatte des Servers.

In den Unterabschnitten dieses Abschnitts werden die einzelnen Schritte näher beschrieben.

6.1.2.1 Initiieren von Aufträgen mittels POST

In diesem Abschnitt werden folgende Schritte beschrieben:

- Senden des Request 284
- JSON-Syntax für POST Requests 284
- Hochladen von Dateien mit dem POST Request 287
- Hochladen von ZIP-Archiven 287

Senden des Request

Ein RaptorXML Server-Auftrag wird mit der HTTP `POST` -Methode initiiert.

HTTP-Methode	URI	Content-Type	Body
`POST`	`http://localhost:8087/v1/queue/`	`application/json`	JSON

Beachten Sie die folgenden Punkte:

- Die obige URI hat eine Server-Adresse, für die die Einstellungen der Anfangskonfiguration[272] verwendet werden.
- Die URI hat einen `/v1/queue/` Pfad, der in der URI vorhanden sein muss. Dabei handelt es sich um einen abstrakten Ordner im Arbeitsspeicher, in den der Auftrag platziert wird.
- Die richtige Versionsnummer `/vN` ist diejenige, die der Server zurückgibt (und nicht notwendigerweise die in der Dokumentation verwendete). Die Nummer, die der Server zurückgibt, ist die Versionsnummer der aktuellen HTTP-Schnittstelle. Frühere Versionsnummern stehen für ältere Versionen der HTTP-Schnittstelle, die weiterhin aus Gründen der Rückwärtskompatibilität unterstützt werden.
- Der Header muss das Feld: `Content-Type: application/json` enthalten. Wenn Sie Dateien allerdings im Body des `POST` Request hochladen möchten, muss der Content-Type des Headers auf `multipart/form-data` (d.h. `Content-Type: multipart/form-data`) gesetzt werden. Nähere Informationen dazu finden Sie im Abschnitt Hochladen von Dateien mit dem POST-Request[287].
- Der Body des Request muss im JSON-Format sein.
- Die zu verarbeitenden Dateien müssen sich auf dem Server befinden, d.h. die Dateien müssen entweder vor Absenden des Request auf den Server kopiert werden oder sie müssen zusammen mit dem POST Request hochgeladen werden[287]. In diesem Fall muss der Content-Type des Message Headers auf `multipart/form-data` gesetzt werden. Nähere Informationen dazu finden Sie im Abschnitt Hochladen von Dateien mit dem POST-Request[287].

Der Request zur Überprüfung der Wohlgeformtheit einer XML-Datei würde im JSON-Format in etwa folgendermaßen aussehen:

```
{
    "command": "wfxml", "args": [ "file:///c:/Test/Report.xml" ]
}
```

Gültige Befehle, ihre Argumente und Optionen sind im Abschnitt zur Befehlszeile[58] dokumentiert.

JSON-Syntax für HTTP POST Requests

```
{
   "command": "Command-Name",
   "options": {"opt1": "opt1-value", "opt2": "opt2-value"},
   "args"   : ["file:///c:/filename1", "file:///c:/filename2"]
}
```

- Der gesamte schwarze Text ist festgelegt und muss inkludiert werden. Dazu gehören alle geschweiften und eckigen Klammern, doppelten Anführungszeichen, Doppelpunkte und Kommas. Whitespaces können normalisiert werden.
- Bei Einträgen in blauer kursiver Schrift handelt es sich um Platzhalter. Sie stehen für Befehlsnamen, Optionen, Optionswerte und Argumentwerte. Eine Beschreibung der einzelnen Befehle finden Sie im Abschnitt zur Befehlszeile[58].

- Die Schlüssel `command` und `args` sind obligatorisch. Der Schlüssel `options` ist optional. Einige `options` Schlüssel haben Standardwerte, d.h. es müssen nur die Optionen angegeben werden, deren Standardwerte geändert werden müssen.
- Alle Strings müssen in doppelte Anführungszeichen gesetzt werden. Boolesche Werte und Zahlen dürfen keine Anführungszeichen haben. D.h.: `{"error-limit": "unlimited"}` und `{"error-limit": 1}` ist korrekt.
- Beachten Sie, dass es sich empfiehlt, anstelle von Dateipfaden Datei-URIs zu verwenden. Dafür werden Schrägstriche verwendet. Bei Verwendung von Windows-Dateipfaden werden umgekehrte Schrägstriche verwendet. Diese Windows Dateipfade müssen in JSON mit Escape versehen werden. (Das Escape-Zeichen ist der umgekehrte Schrägstrich: `"c:\\dir\\filename"`). Beachten Sie, dass URIs und Dateipfade Strings sind, die in Anführungszeichen gesetzt werden müssen.

Hier sehen Sie ein Beispiel mit Optionen. Beachten Sie, dass einige Optionen (wie `input` oder `xslt-version`) einen direkten Optionswert erhalten, während andere (wie `param`) ein Schlüsselwertpaar als Wert erhalten, wofür eine andere Syntax erforderlich ist.

```
{
   "command": "xslt",
   "args": [
           "file:///C:/Work/Test.xslt"
           ],
   "options": {
              "input": "file:///C:/Work/Test.xml",
              "xslt-version": "1",
              "param": {
                       "key": "myTestParam",
                       "value": "SomeParamValue"
                       },
              "output": "file:///C:/temp/out2.xml"
              }
}
```

Im unten gezeigten Beispiel sehen Sie eine dritte Art von Option: die eines Werte-Array (wie bei der Option `xsd` unten). In diesem Fall muss als Syntax die eines JSON Array verwendet werden.

```
{
   "command": "xsi",
   "args": [
           "file:///C:/Work/Test.xml"
           ],
   "options": {
              "xsd" : ["file:///C:/Work/File1.xsd", "file:///C:/Work/File2.xsd"]
              }
}
```


Hochladen von Dateien mit dem POST Request

Zu verarbeitende Dateien können im Body des `POST` Request hochgeladen werden. In diesem Fall muss der `POST` Request folgendermaßen erstellt werden.

Request Header

Setzen Sie im Request Header `Content-Type: multipart/form-data` und definieren Sie einen beliebigen String als Begrenzung. Hier sehen Sie einen Beispiel-Header:

```
Content-Type: multipart/form-data; boundary=---PartBoundary
```

Mit der Begrenzung (MyBoundary) werden die verschiedenen Formulardatenteile im Request Body voneinander abgegrenzt. (*siehe unten*).

Request Body: Message-Teil

Der Body des Request hat die folgenden Formulardatenteile, die durch den im Request Header definierten Begrenzungsstring (*siehe oben*) begrenzt sind:

- *Obligatorische Formulardatenteile:* **`msg`**, welches die angeforderte Verarbeitungsaktion definiert, und **`args`**, welches die Dateien enthält, die als das/die Argument(e) des im `msg` Formulardatenteil definierten Befehls hochzuladen ist/sind. *Siehe Codefragment unten.*
- *Optionaler Formulardatenteil:* Ein Formulardatenteil namens **`additional-files`**, welcher Dateien enthält, die von Dateien in den Formulardatenteilen `msg` oder `args` referenziert werden. Zusätzlich dazu können Formulardatenteile, die nach einer Option des Befehls benannt sind, ebenfalls hochzuladende Dateien enthalten.

Anmerkung: Alle hochzuladenden Dateien werden in einem einzigen virtuellen Verzeichnis angelegt.

In Beispiel-1 (mit Anmerkungen): XML validieren 288 sowie in Beispiel-2: Suchen des Schemas über einen Katalog 289 finden Sie eine genaue Erklärung zum Code.

Testen mit CURL

Sie können den POST Request mit Hilfe einer Datenübertragungsapplikation wie CURL (http://curl.haxx.se/) testen. CURL bietet eine hilfreiche Trace-Option, die die Abgrenzungen der Requests generiert und auflistet, so dass Sie diese nicht mehr manuell erstellen müssen. Eine Anleitung zur Verwendung von CURL finden Sie im Abschnitt Testen mit CURL 292.

Hochladen von ZIP-Archiven

ZIP-Archive können auch hochgeladen werden und Dateien innerhalb eines ZIP-Archivs können über das **`additional-files-`**Schema referenziert werden. Zum Beispiel:

```
additional-files:///mybigarchive.zip%7Czip/biginstance.xml
```

Anmerkung: Der `|zip/`-Teil muss als `%7Czip/` URI-maskiert werden, damit er der URI RFC entspricht, da das Pipe-Symbol `|` direkt nicht zulässig ist. Auch globale Muster wie (`*` und `?`) können verwendet werden. Sie können daher alle XML-Dateien in einem ZIP-Archiv mit einen Befehl wie dem folgenden validieren:

```
{"command": "xsi", "args": ["additional-files:///mybigarchive.zip%7Czip/*.xml"], "options":
{...}}
```

Eine Liste des Beispielcodes finden Sie unter Beispiel-3: Verwendung von ZIP-Archiven 290.

6.1.2.1.1 Beispiel-1 (mit Anmerkungen): XML validieren

Unten sehen Sie das Codefragment des Body eines `POST` Request. Die nummerierten Beschreibungen dazu finden Sie unterhalb. Der im Codefragment-Request gesendete Befehl hätte das folgende Befehlszeilenäquivalent:

```
raptorxml xsi First.xml Second.xml --xsd=Demo.xsd
```

Angefordert wird die Validierung von zwei XML-Dateien anhand eines Schemas. Der Body des Request würde in etwa folgendermaßen aussehen, wenn im Header `---PartBoundary` als Begrenzungsstring definiert wurde (*siehe* *Request Header* 287 weiter oben).

```
-----PartBoundary                                                                    1
Content-Disposition: form-data; name="msg"
Content-Type: application/json

{"command": "xsi", "options": {} , "args": []}                                       2

-----PartBoundary                                                                    3
Content-Disposition: attachment; filename="First.xml"; name="args"
Content-Type: application/octet-stream

   <?xml version="1.0" encoding="UTF-8"?>                                            4
   <test xsi:noNamespaceSchemaLocation="Demo.xsd"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">42</test>

-----PartBoundary                                                                    5
Content-Disposition: attachment; filename="Second.xml"; name="args"
Content-Type: application/octet-stream

   <?xml version="1.0" encoding="UTF-8"?>                                            6
   <test xsi:noNamespaceSchemaLocation="Demo.xsd"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">35</test>

-----PartBoundary                                                                    7
```



```
Content-Disposition: attachment; filename="Demo.xsd"; name="additional-files"
Content-Type: application/octet-stream

   <?xml version="1.0" encoding="UTF-8"?>                                    8
   <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
   elementFormDefault="qualified" attributeFormDefault="unqualified">
      <xs:element name="test" type="xs:int"/>
   </xs:schema>

-----PartBoundary--                                                          9
```

1 Der Name der Begrenzungen für die Hauptformulardatenteile ist im Request Header[287] deklariert. Das Trennzeichen für die Teile muss ein eindeutiger String sein, der nirgends im eingebetteten Dokument vorkommt. Es hat zwei Bindestriche vorangestellt und dient zum Trennen der verschiedenen Teile. Der erste Formulardatenteil ist (in diesem Beispiel) **msg**. Beachten Sie, dass der Content-Type `application/json` ist.

2 Dies ist die Standardsyntax für HTTP POST Requests[285]. Wenn `args` eine Referenz auf eine Datei enthält und wenn weitere Dateien hochgeladen werden, werden beide Dateigruppen an den Server übergeben.

3 Das erste Mitglied des `args` Array ist ein Dateianhang mit dem Namen `First.xml`.

4 Der Text der Datei `First.xml`. Er enthält eine Referenz auf ein Schema mit dem Namen `Demo.xsd`, welches ebenfalls - im Formulardatenteil **additional-files** - hochgeladen wird.

5 Das zweite Mitglied des `args` Array ist ein Anhang mit dem Namen `Second.xml`.

6 Der Text der Datei `Second.xml`. Auch dieser Teil enthält eine Referenz auf das Schema `Demo.xsd`. *Siehe Beschreibungstext 7.*

7 Der erste Zusatzdatenteil enthält die Metadaten für den Anhang `Demo.xsd` .

8 Der Text der Datei `Demo.xsd`.

9 Das Ende des Zusatzdatenteils `Demo.xsd` und des Formulardatenteils `additional-files`. Beachten Sie, dass vor und hinter dem Trennzeichen für den letzten Teil jeweils zwei Bindestriche vorhanden sind.

6.1.2.1.2 Beispiel-2: Suchen des Schemas über einen Katalog

In diesem Beispiel wird zum Suchen des von den zu validierenden XML-Dateien referenzierten XML-Schemas eine Katalogdatei verwendet.

```
-----PartBoundary
Content-Disposition: form-data; name="msg"
Content-Type: application/json

{"command": "xsi", "args": ["additional-files:///First.xml", "additional-
files:///Second.xml"], "options": {"user-catalog": "additional-files:///catalog.xml"}}
```

```
-----PartBoundary
Content-Disposition: attachment; filename="First.xml"; name="additional-files"
Content-Type: application/octet-stream

<?xml version="1.0" encoding="UTF-8"?>
<test xsi:noNamespaceSchemaLocation="http://example.com/Demo.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">42</test>

-----PartBoundary
Content-Disposition: attachment; filename="Second.xml"; name="additional-files"
Content-Type: application/octet-stream

<?xml version="1.0" encoding="UTF-8"?>
<test xsi:noNamespaceSchemaLocation="http://example.com/Demo.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">35</test>

-----PartBoundary
Content-Disposition: attachment; filename="Demo.xsd"; name="additional-files"
Content-Type: application/octet-stream

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
   <xs:element name="test" type="xs:int"/>
</xs:schema>

-----PartBoundary
Content-Disposition: attachment; filename="catalog.xml"; name="additional-files"
Content-Type: application/octet-stream

<?xml version='1.0' encoding='UTF-8'?>
<catalog xmlns='urn:oasis:names:tc:entity:xmlns:xml:catalog'
xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xsi:schemaLocation='urn:oasis:names:tc:entity:xmlns:xml:catalog Catalog.xsd'>
                  <uri name="http://example.com/Demo.xsd" uri="additional-
files:///Demo.xsd"/>
</catalog>

-----PartBoundary--
```

6.1.2.1.3 Beispiel-3: Verwenden von ZIP-Archiven

Auch ZIP-Archive können hochgeladen werden und Dateien innerhalb eines ZIP-Archivs können mit Hilfe des `additional-files`-Schemas referenziert werden. Zum Beispiel:

```
additional-files:///mybigarchive.zip%7Czip/biginstance.xml
```

Anmerkung: Der `|zip/`-Teil muss als `%7Czip/` URI-maskiert werden, damit er der URI RFC entspricht, da das Pipe-Symbol `|` direkt nicht zulässig ist. Auch globale Muster wie (`*` und `?`) können verwendet werden. Sie können daher alle XML-Dateien in einem ZIP-Archiv mit einen Befehl wie dem folgenden validieren:

```
{"command": "xsi", "args": ["additional-files:///mybigarchive.zip%7Czip/*.xml"], "options":
{…}}
```


Anmerkung: Zusätzlich zu `'Content-Disposition: attachment'` ist auch `'Content-Disposition: form-data'` ist gültig. Da mehrere Tools `form-data` als Content Disposition generieren, wird auch der Wert `form-data` als gültig akzeptiert.

Beispiel: Validieren aller XML-Dateien in einem ZIP-Archiv

In diesem Beispiel wird davon ausgegangen, dass alle Schemareferenzen relative Pfade sind und dass sich alle Schemas innerhalb des ZIP-Archivs befinden.

```
-----PartBoundary
Content-Disposition: form-data; name="msg"
Content-Type: application/json

{"command": "xsi", "args": ["additional-files:///Demo.zip%7Czip/*.xml"], "options": {}}

-----PartBoundary
Content-Disposition: attachment; filename="Demo.zip"; name="additional-files"
Content-Type: application/octet-stream

Binary content of Demo.zip archive

-----PartBoundary--
```

Beispiel: Validieren von XML-Dateien in einem ZIP-Archiv, das Referenzen zu externen Schemas enthält

In diesem Beispiel werden die XML-Dateien in einem ZIP-Archiv mittels Referenzen zu einem externen Schema, das in einem zweiten ZIP-Archiv bereitgestellt wird, validiert.

```
-----PartBoundary
Content-Disposition: form-data; name="msg"
Content-Type: application/json

{"command": "xsi", "args": ["additional-files:///Instances.zip%7Czip/*.xml"], "options":
{"user-catalog": "additional-files:///Schemas.zip%7Czip/catalog.xml"}}

-----PartBoundary
Content-Disposition: attachment; filename="Instances.zip"; name="additional-files"
Content-Type: application/octet-stream

Binary content of Instances.zip archive

-----PartBoundary
Content-Disposition: attachment; filename="Schemas.zip"; name="additional-files"
Content-Type: application/octet-stream

Binary content of Schemas.zip archive

-----PartBoundary--
```

6.1.2.1.4 Testen mit CURL

Die Drittanbieter-Applikation CURL (http://curl.haxx.se/) ist ein Befehlszeilentool, mit dem Sie den `POST` Request testen können. CURL bietet eine sehr nützliche Trace-Option, die die Begrenzungen der einzelnen Teile von Requests generiert und auflistet und die sie direkt in Ihren Requests oder als Referenz verwendetn können.

Unten finden Sie ein Beispieltestszenario, in dem eine XML-Datei anhand eines XML-Schemas validiert wird. Für das Beispiel gelten die folgenden Annahmen:

- die unten angeführten Befehle werden von dem Ordner aus ausgeführt, in dem sich die zu validierenden Dateien befinden; (dadurch können einfache relative Pfade zu diesen Dateien verwendet werden). Wenn Sie XMLSpy von Altova installiert haben, finden Sie die in diesem Beispiel verwendeten Dateien im Ordner `Examples` der Applikation. Der vollständige Pfad zu diesem Ordner lautet: `C:\Users\<username>\Documents\Altova\XMLSpy2025\Examples`
- RaptorXML Server wird lokal am Port 8087 ausgeführt

Nähere Informationen zu den CURL-Befehlszeilenoptionen finden Sie in der Hilfe zu CURL.

Aufrufen von CURL mit dem Validierungsbefehl (unter Windows)

```
[input: powershell]
    \path\to\curl.exe -F 'msg={\"command\": \"xsi\", \"args\":[\"additional-
    files:///PurchaseOrder.zip%7Czip/ipo.xml\"], \"options\":{}};type=application/json' -F
    "additional-files=@PurchaseOrder.zip;type=application/octet-stream"
    http://localhost:8087/v1/queue
```

Anmerkung: Wenn in Powershell Anführungszeichen innerhalb von Anführungszeichen stehen, so müssen verschiedene Arten von Anführungszeichen (einfache/doppelte) verwendet werden.

```
[input: cmd]
    \path\to\curl.exe -F "msg={\"command\": \"xsi\", \"args\":[\"additional-
    files:///PurchaseOrder.zip%7Czip/ipo.xml\"], \"options\":{}};type=application/json" -F
    "additional-files=@PurchaseOrder.zip;type=application/octet-stream"
    http://localhost:8087/v1/queue

[output]
    {"jobid": "058F9E97-CB95-43EF-AC0A-496CD3AC43A3", "result": "/v1/results/058F9E97-CB95-
    43EF-AC0A-496CD3AC43A3"}
```

Abrufen des Ergebnisses über die URL von "result"

```
[input]
    \path\to\curl.exe http://localhost:8087/v1/results/058F9E97-CB95-43EF-AC0A-496CD3AC43A3

[output]
```

```
{"jobid":"058F9E97-CB95-43EF-AC0A-496CD3AC43A3","state":"OK","error":{},"jobs":
[{"file":"additional-files:///PurchaseOrder.zip%7Czip/ipo.xml","jobid":"D4B91CB0-CF03-
4D29-B563-B6506E123A06","output":{},"state":"OK","error":{}}]}
```


Die Trace-Option von CURL

CURL hat eine Trace-Option (`--trace-ascii`), die den Verlauf des HTTP-Datenaustausches mit dem Server protokolliert. Die Option ist äußerst nützlich, da sie die zum Initiieren von Aufträgen mit POST benötigten Begrenzungen auflistet. Sie können die Informationen im Trace-Protokoll entweder direkt oder als Referenz verwenden, um die Begrenzungen zu erstellen. Im Codefragment unten sehen Sie das Trace-Protokoll, das bei Ausführung des obigen Befehls aufgezeichnet wurde.

Trace listing

```
== Info:   Trying ::1...
== Info: Connected to localhost (::1) port 8087 (#0)
=> Send header, 217 bytes (0xd9)
0000: POST /v1/queue HTTP/1.1
0019: Host: localhost:8087
002f: User-Agent: curl/7.42.1
0048: Accept: */*
0055: Content-Length: 2939
006b: Expect: 100-continue
0081: Content-Type: multipart/form-data; boundary=--------------------
00c1: ----d887ed58324015c3
00d7:
<= Recv header, 23 bytes (0x17)
0000: HTTP/1.1 100 Continue
=> Send data, 393 bytes (0x189)
0000: --------------------------d887ed58324015c3
002c: Content-Disposition: form-data; name="msg"
0058: Content-Type: application/json
0078:
007a: {"command": "xsi", "args":["additional-files:///PurchaseOrder.zi
00ba: p%7Czip/ipo.xml"], "options":{}}
00dc: --------------------------d887ed58324015c3
0108: Content-Disposition: form-data; name="additional-files"; filenam
0148: e="PurchaseOrder.zip"
015f: Content-Type: application/octet-stream
0187:
=> Send data, 2498 bytes (0x9c2)
0000: PK........"..6}.c.....M.......ipo.xsd.T.N.@.}N....O 5v.}..S....(
0040:  .JU/...$Y..5{.E.•.....I*...g...Y...\....Z..~.......P.A.ct....•y.
...
0940: ........."..6]g......l.............. .......address.xsdPK.......
0980: ..."..6I..v.................. .......ipo.xmlPK..................
09c0: ..
=> Send data, 48 bytes (0x30)
0000:
0002: --------------------------d887ed58324015c3--
<= Recv header, 22 bytes (0x16)
0000: HTTP/1.1 201 Created
<= Recv header, 13 bytes (0xd)
0000: Allow: POST
```

```
<= Recv header, 32 bytes (0x20)
0000: Content-Type: application/json
<= Recv header, 37 bytes (0x25)
0000: Date: Fri, 24 Jul 2015 16:58:08 GMT
<= Recv header, 24 bytes (0x18)
0000: Server: CherryPy/3.6.0
<= Recv header, 21 bytes (0x15)
0000: Content-Length: 111
<= Recv header, 2 bytes (0x2)
0000:
<= Recv data, 111 bytes (0x6f)
0000: {"jobid": "058F9E97-CB95-43EF-AC0A-496CD3AC43A3", "result": "/v1
0040: /results/058F9E97-CB95-43EF-AC0A-496CD3AC43A3"}
== Info: Connection #0 to host localhost left intact
```

Anmerkung: Anhand des obigen Codefragments sehen Sie, dass zusätzlich zu `'Content-Disposition: attachment'` auch `'Content-Disposition: form-data'` gültig ist.

Aufrufen von CURL mit dem Befehl zur Überprüfung der Wohlgeformtheit (unter Linux)

```
/path/to/curl -F 'msg={"command": "wfxml", "args":[]};type=application/json' -F
"args=@ipo.xml;type=application/octet-stream" http://localhost:8087/v1/queue

/path/to/curl -F 'msg={"command": "wfxml", "args":["additional-files:///ipo.zip%
7Czip/ipo.xml"]};type=application/json' -F "additional-
files=@ipo.zip;type=application/octet-stream" http://localhost:8087/v1/queue
```

6.1.2.1.5 Beispiel-6: XQuery-Ausführung

In diesem Beispiel wird unter Windows mittels Power Shell ein XQuery-Dokument an einem XML-Dokument ausgeführt. Beide Dokumente befinden sich im Ordner **`examples`** Ihres Applikationsordners (RaptorXMLServer2025).

Anmerkung: Unter Umständen werden Anführungszeichen auf anderen Shells anders verwendet ('bash' funktioniert mit dem Beispiel, wenn man anstelle von 'curl.exe' 'curl' verwendet).

Bereitstellen des POST-Requests zur Validierung von Inline XBRL mittels CURL

Unten finden Sie einen CURL-Beispielbefehl, mit dem ein Inline XBRL-Validierungs-Request gesendet wird.

```
\path\to\curl.exe -F 'msg={\"command\": "xquery", "args": ["additional-
files:///CopyInput.xq"], "options": {"input": "additional-files:///simple.xml",
"output": "MyQueryResult"}};type=application/json' -F "additional-
files=@CopyInput.xq;type=text/plain"  -F "additional-
files=@simple.xml;type=application/xml" http://localhost:8087/v1/queue
```

Zur einfacheren Lesbarkeit:

```
(1) -F 'msg={
(2)     "command": "xquery",
(3)     "args": ["additional-files:///CopyInput.xq"],
(4)     "options": {"input": "additional-files:///simple.xml", "output":
"MyQueryResult"}
(5) };type=application/json'
(6) -F "additional-files=@CopyInput.xq;type=text/plain"
(7) -F "additional-files=@simple.xml;type=application/xml"
(7) http://localhost:8087/v1/queue
```

Input
Die verschiedenen Teile des CURL-Befehls werden unten anhand der oben angeführten Nummern erklärt.

(1) **`-F 'msg={...}'`** definiert ein Formularfeld mit dem Namen **`'msg'`**

Aufgrund der Option **`-F`** (i) generiert CURL ein mehrteiliges Formular-POST mit dem **`Content-Type: multipart/form-data`** und (ii) fügt dieses form-Feld automatisch zum Request Header hinzu. Zur Beschreibung des von RaptorXML Server auszuführenden Befehls verwenden wir ein JSON-Objekt.

```
Content-Type: multipart/form-data; boundary=--------------------
```

CURL übersetzt diese Option im HTTP-Request folgendermaßen:

```
Content-Disposition: form-data; name="msg"
Content-Type: application/json
{"command": "xquery", "args": ["additional-files:///CopyInput.xq"], "options": {"input":
"additional-files:///simple.xml", "output": "MyQueryResult"}}
```

(2) Der auf dem Server auszuführende RaptorXML Server-Befehl. Informationen zu Befehlen, die hier verwendet werden können, finden Sie unter Befehlszeilenschnittstelle (CLI)[58]. Der Befehl für die XQuery-Ausführung lautet in unserem Beispiel **`XQuery`**[98].

(3) Die Argumente des Befehls (die von der RaptorXML Server-Befehlszeile akzeptiert werden) sind als JSON-Array kodiert. RaptorXML Server verwendet ein explizites Schema **`additional-files://`**, um zusätzliche Ressourcen in einem separaten **`additional-files`**-Formularfeld zu referenzieren. In unserem Beispiel referenzieren wir das XQuery-Dokument **`CopyInput.xq`**.

Anmerkung: Alle Ressourcen im **`args`**-Array müssen auf dem Server verfügbar sein oder mit einem Request ähnlich denen in Punkt (6) und (7) bereitgestellt werden.

(4) Die Optionen des Befehls (die von der RaptorXML Server-Befehlszeile akzeptiert werden) sind als JSON-Objekt kodiert. Wenn die Standardwerte von Optionen für Ihre Bedürfnisse geeignet sind (siehe Abschnitt

CLI[58]), kann dieser Teil weggelassen werden. In unserem Beispiel definieren wir (i) die XML-Datei, an der die XQuery ausgeführt werden soll und (ii) die Datei, in der die Ausgabe der XQuer-Ausführung gespeichert werden soll.

(5) Der Content-Type des **msg**-Formularfelds wird nach der Definition des Formularfelds angegeben und durch ein Semikolon vom Wert des Formularfelds getrennt. In unserem Beispiel wird der Content Type von **msg** durch **type=application/json** angegeben.

(6) Dateien, die zusätzliche Ressourcen für den Befehl enthalten, können mit Hilfe des **additional-files**-Formularfelds definiert werden. In unserem Beispiel definieren wir zwei zusätzliche Ressourcen: (i) **@CopyInput.xq**, gefolgt von einem Semikolon als Trennzeichen und anschließend seinem Content-Type, den wir als **type=text/plain** angeben; (i) **simple.xml**, gefolgt von einem Semikolon als Trennzeichen und anschließend seinem Content-Type, den wir als **type=application/xml** angeben.

Anmerkung: Stellen Sie dem Dateinamen das Zeichen **@** voran, damit CURL (i) den Dateinamen als Wert der Eigenschaft **filename** und (ii) den Inhalt der Datei als Wert des Formulars verwendet. Das additional-files-Formularfeld kann mehrmals bereitgestellt werden, einmal für jede zusätzliche für den Befehl erforderliche Ressource. CURL übersetzt diese Option in den folgenden HTTP-Request:

```
Content-Disposition: form-data; name="additional-files"; filenam
Content-Type: text/plain
<<content of CopyInput.xq>>

Content-Disposition: form-data; name="additional-files"; filename="simple.xml"
Content-Type: application/xml
<<content of simple.xml>>
```

Anmerkung: Dateien aus anderen Ordnern können durch Voranstellen des relativen Pfads vor den Dateinamen angegeben werden: `-F "additional-files=@Examples/CopyInput.xq;type=text/plain"`. Wenn jedoch eine zusätzliche Datei aus einem anderen Ordner auf diese Art angegeben wird, muss sie durch Angabe des Dateinamens allein referenziert werden. Beispiel:

```
\path\to\curl.exe -F 'msg={\"command\": "xquery", "args": ["additional-
files:///CopyInput.xq"], "options": {"output":
"MyQueryResult"}};type=application/json' -F "additional-
files=@Examples/CopyInput.xq;type=text/plain"  http://localhost:8087/v1/queue
```

Wenn Sie eine Ordnerstruktur beibehalten möchten, speichern Sie die Dateien in einem ZIP-Ordner und referenzieren Sie die Dateien auf die übliche Art für ZIP-Ordner[290].

Ausgabe
Von RaptorXML Server wird ein JSON-Objekt ausgegeben:

```
{"jobid": "42B8A75E-0180-4E05-B28F-7B46C6A0C686", "result": "/v1/results/42B8A75E-0180-
4E05-B28F-7B46C6A0C686"}
```

Das JSON-Objekt enthält einen **jobid**-Schlüssel und einen **result**-Schlüssel. Der Wert des **result**-Schlüssels ist der Pfad zum Ergebnis. Dieser Pfad muss an den Teil **<scheme>://<host>:<port>**, der zum

Bereitstellen des Request verwendet wird, angehängt werden. In unserem Beispiel würde die vollständige Ergebnis-URL folgendermaßen lauten: **http://localhost:8087/v1/results/42B8A75E-0180-4E05-B28F-7B46C6A0C686**. Die Ergebnis-URL wird auch verwendet, um das Ergebnis der Befehlsausführung anzufordern. *Siehe* *Abrufen des Ergebnisdokuments* 301.

Abrufen des Fehlers/der Nachricht/der Ausgabe des POST-Request

Der Input-Befehl zum Abrufen des Fehlers/der Nachricht/der Ausgabe des POST-Request (*siehe* *Abrufen von Fehler-/Meldungs-/Ausgabedokumenten* 304) würde in etwa folgendermaßen lauten:

```
curl.exe http://localhost:8087/v1/results/42B8A75E-0180-4E05-B28F-7B46C6A0C686
```

In our example, this command returns the following JSON object:

```
{"jobid":"42B8A75E-0180-4E05-B28F-7B46C6A0C686","state":"OK","error":{},"jobs":
[{"file":"additional-files:///simple.xml","jobid":"768656F9-F4A1-4492-9676-
C6226E30D998","output":{"result.trace_file":["/v1/results/768656F9-F4A1-4492-9676-
C6226E30D998/output/trace.log"],"xquery.main_output_files":["/v1/results/768656F9-F4A1-
4492-9676-C6226E30D998/output/1"],"xquery.additional_output_files":
[]},"state":"OK","output-mapping":{"/v1/results/768656F9-F4A1-4492-9676-
C6226E30D998/output/1":"file:///C:/ProgramData/Altova/RaptorXMLXBRLServer2016/Output/768
656F9-F4A1-4492-9676-C6226E30D998/MyQueryResult"},"error":{}}]}
```

Zur besseren Lesbarkeit wurde dies unten in separaten Zeilen formatiert und mit nummerierten Beschreibungen versehen:

```
%1 :
(2)    "jobid":"42B8A75E-0180-4E05-B28F-7B46C6A0C686",
(3)    "state":"OK",
(4)    "error":{},
(5)    "jobs":[{
(6)       "file":["additional-files:///simple.xml"],
(7)       "jobid":"768656F9-F4A1-4492-9676-C6226E30D998",
(8)       "output":{
(9)          "result.trace_file":["/v1/results/768656F9-F4A1-4492-9676-
C6226E30D998/output/trace.log"],
(10)         "xquery.main_output_files":["/v1/results/768656F9-F4A1-4492-9676-
C6226E30D998/output/1"],
(11)         "xquery.additional_output_files":[]},
(12)         "state":"OK",
(13)         "output-mapping":{
(14)            "/v1/results/768656F9-F4A1-4492-9676-C6226E30D998/output/1":
(15)            "file:///C:/ProgramData/Altova/RaptorXMLXBRLServer2016/Output/768656F9-
F4A1-4492-9676-C6226E30D998/MyQueryResult"
(16)          },
(17)       "error":{}
(18)    }]
(19) }
```

Im Folgenden finden Sie eine Erklärung dieser Liste:

(1) Das Ergebnis wird als JSON-Objekt zurückgegeben.

(2) Die Auftrags-ID auf der ersten Ebene ist der Auftragshaupt-Identifier.

(3) Der Status für diesen Auftrag ist OK. Mögliche Zustände sind: *none; Dispatched (abgesendet); Running (wird ausgeführt); Canceled (abgebrochen); Crashed (abgestürzt); OK; Failed (fehlgeschlagen)*.

(4) Das JSON-Fehlerobjekt in unserem Beispiel ist leer. Es kann die von RaptorXML Server ausgegebene JSON-Serialisierung des Fehlers enthalten.

(5) Der Hauptauftrag (auf der ersten Ebene) generiert Unteraufträge (z.B. einen pro Argument).

(6) Das Argument für diesen Auftrag ist die XML-Instanzdatei: **`additional-files:///simple.xml`**.

(7) Unteraufträge haben ebenfalls eine Auftrags-ID, anhand welcher der Status abgefragt oder die Ergebnisse abgerufen werden können. Die Auftragsausführung ist asynchron. Aufgrund dessen können kurze Aufträge, die nach längeren Aufträgen gesendet wurden, eventuell früher fertig sein.

(8) bis (16) Das JSON **`output`**-Objekt enthält Schlüssel für die Server-generierten Ausgabedateien, die über HTTP abgerufen werden können. Einige Schlüssel (wie z.B. **`xquery.main_output_files`**) definieren URLs zu den auf dem Server gespeicherten generierten Dateien. Diese lokalen Pfade auf dem Server können auf Namen gemappt werden, die in HTTP-URLs als JSON **`output-mapping`**-Objekte verwendet werden können. Solche URLs dienen zum Abrufen von Ausgabedateien über HTTP und setzen sich folgendermaßen zusammen:

```
<scheme>://<host>:<port>/<output-mapping-value>
```

Unser Beispiel zum Abrufen der XQuery-Hauptausgabedatei würde daher folgendermaßen aussehen:

```
curl.exe http://localhost:8087/v1/results/768656F9-F4A1-4492-9676-C6226E30D998/output/1
```

Beachten Sie, dass der erste Wert (14) im **`output-mapping`**-Objekt (13), der Mapping-Wert ist, den wir mit dem Schlüssel an die XQuery-Ausgabe gebunden haben **`file:///C:/ProgramData/Altova/RaptorXMLXBRLServer2016/Output/768656F9-F4A1-4492-9676-C6226E30D998/MyQueryResult`** Wir können die Datei dadurch über den Mapping-Wert referenzieren.

6.1.2.2 Server-Antwort auf den POST Request

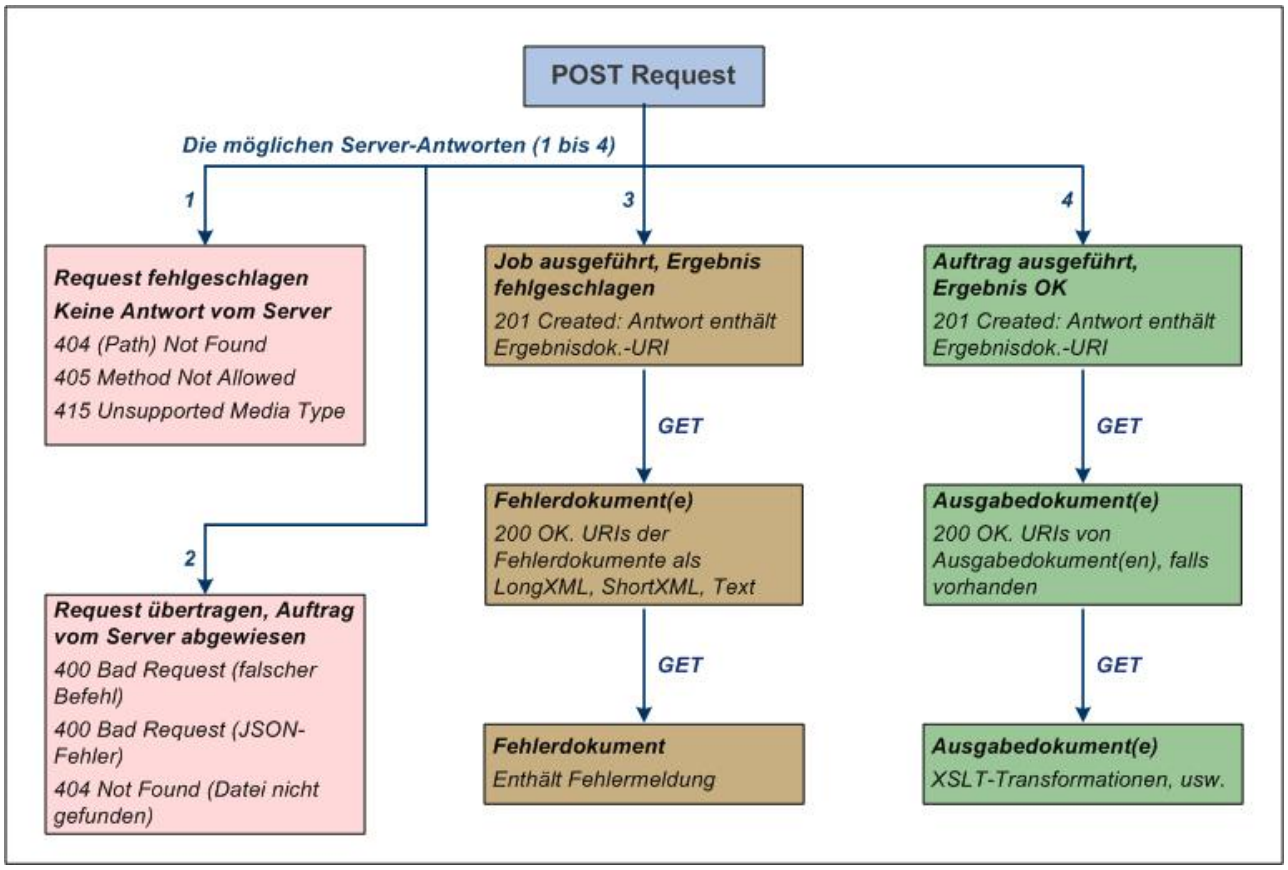
In diesem Abschnitt werden folgende Schritte beschrieben:

- Übersicht über mögliche Server-Antworten 298
- Antwort: Request fehlgeschlagen, keine Antwort vom Server 299
- Antwort: Der Request wurde übertragen, doch der Auftrag wurde vom Server abgewiesen 300
- Antwort: Der Auftrag wurde ausgeführt (mit positivem oder negativem Ergebnis) 300

Wenn ein `POST` Request erfolgreich an den Server bereitgestellt wurde, wird der Auftrag in die Server-Warteschlange platziert. Der Server gibt eine `201 Created` Meldung und eine Ergebnisdokument-URI zurück. Der Auftrag wird zum frühestmöglichen Zeitpunkt verarbeitet. Falls in der Zwischenzeit das Ergebnisdokument angefordert wird 301, wird eine Meldung `"status": "Running"` zurückgegeben, wenn der Auftrag gestartet, aber noch nicht fertig gestellt wurde; der Client sollte das Dokument zu einem späteren Zeitpunkt wieder

anfordern. Mit dem Status `Dispatched` wird angegeben, dass sich der Auftrag in der Server-Warteschlange befindet, aber noch nicht gestartet wurde.

Das Ergebnis des Auftrags (z.B. einer Validierungsanforderung) kann negativ (Validierung fehlgeschlagen) oder positiv (Validierung erfolgreich) sein. In beiden Fällen wird eine `201 Created` Meldung zurückgegeben und ein Ergebnisdokument wird generiert. Es kann auch sein, dass der `POST` Request nicht an den Server übertragen wurde (*Request fehlgeschlagen*) oder der Request zwar übertragen wurde, der Auftrag vom Server aber abgewiesen wurde (*Request übertragen, doch Auftrag abgewiesen*). Im nachstehenden Diagramm sind die verschiedenen möglichen Ergebnisse dargestellt.



Request fehlgeschlagen, keine Antwort vom Server

Wenn Requests nicht erfolgreich an den Server übertragen werden können, sind die häufigsten Fehler die folgenden:

Meldung	Erklärung
404 Not Found	Der richtige Pfad lautet:`http://localhost:8087/v1/queue/`
405 Method Not Allowed	Die angegebene Methode ist für diese Ressource unzulässig. Verwenden Sie die `POST` Methode.
415 Unsupported Media Type	Der Message Header sollte lauten: `Content-Type:application/json`.

Request übertragen, doch der Auftrag wurde vom Server abgewiesen

Wenn Requests erfolgreich an den Server übertragen wurden, kann der Server diese aus einem der folgenden Gründe abweisen:

Message	Erklärung
400 Bad Request (*bad cmd*)	Der RaptorXML Befehl 58 ist falsch.
400 Bad Request (*json error*)	Der Request Body enthält einen JSON-Syntaxfehler 285.
404 File Not Found	Überprüfen Sie die Datei-URI (oder die Dateipfad)-Syntax 285 aller im Befehl genannten Dateien.

Der Auftrag wurde ausgeführt (mit positivem oder negativem Ergebnis)

Wenn ein Auftrag (z.B. ein Validierungsauftrag) ausgeführt wird, kann das Ergebnis positiv (*OK*) oder negativ (*fehlgeschlagen*) sein. So kann z.B. das Ergebnis eines Validierungsauftrags positiv (*OK*) sein, wenn das zu validierende Dokument gültig ist und negativ (*fehlgeschlagen*), wenn das Dokument ungültig ist.

In beiden Fällen wurde der Auftrag ausgeführt, allerdings mit unterschiedlichen Ergebnissen. In beiden Fällen wird eine `201 Created` Meldung zurückgegeben, sobald der Auftrag erfolgreich in die Warteschlange aufgenommen wurde. In beiden Fällen wird auch eine Ergebnisdokument-URI an den HTTP-Client, von dem der Request stammt, zurückgegeben. Nachdem das Ergebnisdokument erstellt wurde, kann es mit einem HTTP `GET` Request abgerufen werden.

Möglicherweise wurde das Ergebnisdokument noch nicht erstellt, wenn mit der Verarbeitung des Auftrags noch nicht begonnen oder der Auftrag noch nicht fertig verarbeitet wurde. Während das Ergebnisdokument während dieser Zeit angefordert wird, wird eine `"status": "Running"` Meldung zurückgegeben, wenn der Auftrag gestartet, aber noch nicht fertig gestellt wurde; mit dem Status `Dispatched` wird angegeben, dass sich der Auftrag in der Server-Warteschlange befindet, aber noch nicht gestartet wurde.

Zusätzlich zum Ergebnisdokument können auch andere Dokumente generiert werden. Dazu gehören die folgenden:

- *Auftrag wurde mit dem Ergebnis 'Fehlgeschlagen' ausgeführt:* Ein Fehlerprotokoll wird in drei Formaten erstellt: Text, langes XML-Dokument und kurzes XML-Dokument. Die URIs dieser drei Dokumente werden im Ergebnisdokument (welches im JSON-Format ist) gesendet. Die URIs können in einem HTTP `GET` Request verwendet werden, um die Fehlerdokumente abzurufen 304.
- *Auftrag wurde mit dem Ergebnis 'OK' ausgeführt:* Der Auftrag wird erfolgreich ausgeführt und die Ausgabedokumente - wie z.B. die durch eine XSLT-Transformation erzeugte Ausgabe - werden erstellt. Wenn Ausgabedateien generiert wurden, werden ihre URIs im Ergebnisdokument, das im JSON-Format ist, gesendet. Die URIs können anschließend in einem HTTP `GET` Request verwendet werden, um die Ausgabedokumente abzurufen. Beachten Sie, dass nicht alle Aufträge Ausgabedateien haben; ein Beispiel dafür ist ein Validierungsauftrag. Außerdem kann es vorkommen, dass ein Auftrag mit dem Status '*OK*' fertig gestellt wurde, dass es aber Warnungen und/oder andere Meldungen gab, die in Fehlerdateien geschrieben wurden. In diesem Fall werden die URIs der Fehlerdateien (zusätzlich zu den Ausgabedokumenten) ebenfalls im Ergebnisdokument gesendet.

Eine Beschreibung dieser Dokumente und wie Sie diese aufrufen, finden Sie unter Abrufen des Ergebnisdokuments 301 und Abrufen von Fehler-/Ausgabedokumenten 304.

6.1.2.3 Abrufen des Ergebnisdokuments

In diesem Abschnitt werden folgende Schritte beschrieben:

Die Ergebnisdokument-URI

Bei jeder Erstellung eines Auftrags wird ein Ergebnisdokument erstellt, unabhängig davon, ob das Ergebnis eines Auftrags (z.B. einer Validierung) positiv (Dokument ist gültig) oder negativ (Dokument ungültig) ist. In beiden Fällen wird eine `201 Created` Meldung zurückgegeben. Diese Meldung ist im JSON-Format und enthält eine relative URI des Ergebnisdokuments. Das JSON-Fragment sieht in etwa folgendermaßen aus:

```
{
   "result": "/v1/results/E6C4262D-8ADB-49CB-8693-990DF79EABEB",
   "jobid": "E6C4262D-8ADB-49CB-8693-990DF79EABEB"
}
```

Das Objekt `result` enthält die relative URI des Ergebnisdokuments. Die URI ist relativ zur Serveradresse 277. Wenn die Serveradresse z.B. `http://localhost:8087/` (die Adresse in der Anfangskonfiguration 272) ist, so lautet die erweiterte URI des im Codefragment oben angegebenen Ergebnisdokuments:

```
http://localhost:8087/v1/results/E6C4262D-8ADB-49CB-8693-990DF79EABEB
```

Anmerkung: Die richtige Versionsnummer `/vN` ist diejenige, die der Server zurückgibt (und nicht notwendigerweise die in der Dokumentation verwendete). Die Nummer, die der Server zurückgibt, ist die Versionsnummer der aktuellen HTTP-Schnittstelle. Frühere Versionsnummern stehen für ältere Versionen der HTTP-Schnittstelle, die jedoch weiterhin aus Gründen der Rückwärtskompatibilität unterstützt werden.

Abrufen des Ergebnisdokuments

Um das Ergebnisdokument abzurufen, senden Sie die erweiterte URI des Dokuments (*siehe oben* 301) in einem HTTP `GET` Request. Daraufhin wird das Ergebnisdokument zurückgegeben. Es könnte eine der unten beschriebenen allgemeinen Arten sein.

Anmerkung: Wenn ein Auftrag erfolgreich in die Server-Warteschlange gestellt wurde, wird die URI des Ergebnisdokuments zurückgegeben. Wenn der Client das Ergebnis anfordert, bevor der Auftrag gestartet wurde (weil er sich noch in der Warteschlange befindet) wird die Meldung `"status": "Dispatched"` zurückgegeben. Wenn der Auftrag gestartet, aber noch nicht fertig gestellt wurde (weil es sich z.B. um einen großen Auftrag handelt), wird die Meldung `"status": "Running"` zurückgegeben. In beiden Fällen sollte der Client einige Zeit warten, bevor er das Ergebnisdokument erneut anfordert.

Anmerkung: In den Beispieldokumenten unten wird immer davon ausgegangen, dass der Zugriff des Client eingeschränkt ist 269. Daher wird angenommen, dass Fehlerdokumente, Meldungsdokumente und Ausgabedokumente im entsprechenden Auftragsverzeichnis auf dem Server gespeichert werden. Die URIs für diese Dokumente im Ergebnisdokument sind daher alle relative URIs. Keine davon ist eine Datei-URI (wie dies

bei unbeschränktem Client-Zugriff[268] der Fall wäre). Nähere Informationen zu diesen URIs finden Sie im Abschnitt Abrufen von Fehler-/Ausgabedokumenten[304].

Ergebnisdokument, das URIs von Fehlerdokumenten enthält
Wenn der angeforderte Auftrag mit dem Status *Fehlgeschlagen* beendet wurde, hat der Auftrag ein negatives Ergebnis zurückgegeben. So wurde z.B. bei einem Validierungsauftrag das Ergebnis: "Dokument ungültig" zurückgegeben. Die bei der Ausführung des Auftrags ausgegebenen Fehler werden in Fehlerprotokollen gespeichert, die in drei Formaten erstellt werden: (i) Text, (ii) long-XML (detailliertes Fehlerprotokoll) und (iii) short-XML (weniger ausführliches Fehlerprotokoll). Siehe JSON-Codefragment unten.

```
{
    "jobid": "6B4EE31B-FAC9-4834-B50A-582FABF47B58",
    "state": "Failed",
    "error":
    {
      "text": "/v1/results/6B4EE31B-FAC9-4834-B50A-582FABF47B58/error/error.txt",
      "longxml": "/v1/results/6B4EE31B-FAC9-4834-B50A-582FABF47B58/error/long.xml",
      "shortxml": "/v1/results/6B4EE31B-FAC9-4834-B50A-582FABF47B58/error/short.xml"
    },
    "jobs":
    [
      {
        "file": "file:///c:/Test/ExpReport.xml",
        "jobid": "20008201-219F-4790-BB59-C091C276FED2",
        "output":
        {
        },
        "state": "Failed",
        "error":
        {
          "text": "/v1/results/20008201-219F-4790-BB59-C091C276FED2/error/error.txt",
          "longxml": "/v1/results/20008201-219F-4790-BB59-C091C276FED2/error/long.xml",
          "shortxml": "/v1/results/20008201-219F-4790-BB59-C091C276FED2/error/short.xml"
        }
      }
    ]
}
```

Beachten Sie bitte die folgenden Punkte:

- Aufträge haben Unteraufträge.
- Fehler, die auf Unterauftragsebene entstehen, setzen sich bis zum Auftrag der obersten Ebene fort. Der Status des Auftrags der obersten Ebene ist nur dann *OK,* wenn alle seine Unteraufträge ebenfalls den Status *OK* haben.
- Jeder Auftrag oder Unterauftrag hat sein eigenes Fehlerprotokoll.
- Fehlerprotokolle enthalten Warnungsprotokolle, d.h. auch wenn ein Auftrag mit dem Status *OK* beendet wird, kann er URIs von Fehlerdateien enthalten.
- Die URIs von Fehlerdateien sind relativ zur Serveradresse (*siehe oben*[301]).

Ergebnisdokument, das URIs von Ausgabedokumenten enthält
Wenn der angeforderte Auftrag mit dem Status *OK* beendet wurde, wurde vom Auftrag ein positives Ergebnis zurückgegeben. So wurde z.B. bei einem Validierungsauftrag das Ergebnis: "Dokument gültig" zurückgegeben. Wenn beim Auftrag ein Ausgabedokument - z.B. das Ergebnis einer XSLT-Transformation - zurückgegeben wurde, so wird die URI des Ausgabedokuments zurückgegeben. Siehe JSON-Codefragment unten.


```
{
    "jobid": "5E47A3E9-D229-42F9-83B4-CC11F8366466",
    "state": "OK",
    "error":
    {
    },
    "jobs":
    [
      {
        "file": "file:///c:/Test/SimpleExample.xml",
        "jobid": "D34B5684-C6FF-4A7A-BF35-EBB9A8A8C2C8",
        "output":
        {
          "xslt-output-file":
          [
            "/v1/results/D34B5684-C6FF-4A7A-BF35-EBB9A8A8C2C8/output/1"
          ]
        },
        "state": "OK",
        "output-mapping":
        {
         "/v1/results/D34B5684-C6FF-4A7A-BF35-EBB9A8A8C2C8/output/1":
         "file:///c:/temp/test.html"
        },
        "error":
        {
        }
      }
    ]
}
```

Beachten Sie die folgenden Punkte:

- Die Ausgabedatei wird im Ordner `output` des Auftrags erstellt. Sie können zum Aufrufen der Datei die relative URI verwenden.
- Die URIs der Ausgabedateien sind relativ zur Serveradresse (*siehe oben* 301).
- Das Datenelement `output-mapping` mappt das Ausgabedokument im Auftragsverzeichnis auf dem Server auf den durch den Client in der Auftragsanforderung angegebenen Ordner. Beachten Sie, dass nur Ausgabedokumente, die vom Client in der Auftragsanforderung definiert sind, ein Mapping haben; Dateien im Zusammenhang mit dem Auftrag, die vom Server generiert werden (wie z.B. Fehlerdateien), haben kein Mapping.
- Alternativ dazu können alle generierten Ergebnisdokumente für einen bestimmten Auftrag über die URL "`/v1/results/`*JOBID*`/output/zip`" als ZIP-Archiv aufgerufen werden. Diese Funktion steht im uneingeschränkten Dateisystemmodus nicht zur Verfügung. Beachten Sie bitte, dass das ZIP-Archiv beschädigte Dateinamen enthält, die mit Hilfe des Objekts `output-mapping` wieder den tatsächlichen Namen zugeordnet werden müssen.

Ergebnisdokument, das keine URI enthält
Wenn der angeforderte Auftrag mit dem Status *OK* beendet wurde, hat der Auftrag ein positives Ergebnis zurückgegeben. So wurde z.B. bei einem Validierungsauftrag das Ergebnis: "Dokument gültig" zurückgegeben. Bei einigen Aufträgen - z.B. bei der Validierung oder der Wohlgeformtheitsprüfung - wird kein Ausgabedokument erzeugt. Wenn ein Auftrag dieser Art mit dem Status *OK* beendet wird, hat das Ergebnisdokument weder die URI eines Ausgabedokuments noch die URI eines Fehlerprotokolls. Siehe JSON-Codefragment unten.

```
{
    "jobid": "3FC8B90E-A2E5-427B-B9E9-27CB7BB6B405",
```

```
    "state": "OK",
    "error":
    {
    },
    "jobs":
    [
      {
        "file": "file:///c:/Test/SimpleExample.xml",
        "jobid": "532F14A9-F9F8-4FED-BCDA-16A17A848FEA",
        "output":
        {
        },
        "state": "OK",
        "error":
        {
        }
      }
    ]
}
```

Beachten Sie die folgenden Punkte:

- Sowohl das Ausgabedokument als auch das Fehlerprotokoll des Unterauftrags im Codefragment oben sind leer.
- Ein Auftrag, der mit dem Status *OK* beendet wurde, kann trotzdem Warnungen oder andere Meldungen enthalten, die in Fehlerdateien protokolliert sind. In diesem Fall enthält das Ergebnisdokument URIs von Fehlerdateien, obwohl der Auftrag mit dem Status *OK* beendet wurde.

Aufrufen von im Ergebnisdokument aufgelisteten Fehler- und Ausgabedokumenten

Fehler- und Ergebnisdokumente können mit HTTP `GET` Requests aufgerufen werden. Eine Beschreibung dazu finden Sie im nächsten Abschnitt Abrufen von Fehler-/Meldungs-/Ausgabedokumenten 304.

6.1.2.4 Abrufen von Fehler-/Meldungs-/Ausgabedokumenten

Ein Ergebnisdokument 301 kann die Datei-URIs oder relativen URIs von Fehlerdokumenten 302, Meldungsdokumenten (wie z.B. Logs) und/oder Ausgabedokumenten 302 enthalten. (In manchen Fällen 303 enthält das Ergebnisdokument keine URI.) Im Folgenden sind die verschiedenen URI-Arten beschrieben 304.

Um diese Dokumente über HTTP aufrufen zu können, gehen Sie folgendermaßen vor:

1. Erweitern Sie die relative URI 305 der Datei im Ergebnisdokument zu ihrer absoluten URI.
2. Verwenden Sie die erweiterte URI in einem HTTP GET Request 306, um die Datei aufzurufen.

URIs von Fehler-/Meldungs-/Ausgabedokumenten (im Ergebnisdokument)

Das Ergebnisdokument enthält URIs von Fehler-, Meldungs- und/oder Ausgabedokumenten. Fehler- und Meldungsdokumente sind Dokumente, die vom Server im Zusammenhang mit einem Auftrag generiert werden; sie werden immer im Auftragsverzeichnis auf dem Server gespeichert. Ausgabedokumente (wie z.B. die Ausgabe von XSLT-Transformationen) können in einem der folgenden Ordner gespeichert werden:

- unter jedem Dateipfad, auf den der Server Zugriff hat. Damit Ausgabedateien in jedem beliebigen Ordner gespeichert werden können, muss der Server für den unbeschränkten Client-Zugriff[274] konfiguriert sein (Standardeinstellung).
- im Auftragsverzeichnis auf dem Server. Der Server ist so konfiguriert[272], dass er nur eingeschränkten Zugriff von Clients gestattet.

Wenn ein Client verlangt, dass eine Ausgabedatei erstellt wird, hängt der Pfad, unter dem die Ausgabedatei gespeichert wird, von der Option `server.unrestricted-filesystem-access`[274] der Server-Konfigurationsdatei ab.

- Wenn unbeschränkter Zugriff besteht, wird die Datei in dem vom Client angegebenen Ordner gespeichert. Die für das Dokument zurückgegebene URI ist in diesem Fall eine Datei-URI.
- Wenn der Zugriff eingeschränkt ist, wird die Datei im Auftragsverzeichnis gespeichert. Ihre URI ist dann eine relative URI. Außerdem gibt es ein Mapping dieser relativen URI auf die vom Client angegebene URL. (*Siehe Codefragment in Ergebnisdokument, das URIs von Ausgabedokumenten enthält*[302])

Zusammenfassend gibt es die folgenden URI-Arten:

Datei-URI von Fehler-/Meldungsdokumenten
Diese Dokumente werden im Auftragsverzeichnis auf dem Server gespeichert. Datei-URIs haben die folgende Form:

```
Datei:///<Root-Ausgabe-Verz>/AUFTRAGSID/meldung.doc
```

Datei-URIs von Ausgabedokumenten
Diese Dokumente können in jedem beliebigen Ordner gespeichert werden. Datei-URIs haben die folgenden Form:

```
Datei:///<Pfad-zur-Datei>/ausgabe.doc
```

HTTP-URI von Fehler-/Meldungs-/Ausgabedokumenten
Diese Dokumente werden im Auftragsverzeichnis auf dem Server gespeichert. URIs sind relativ zur Server-Adresse und müssen zur vollständigen HTTP-URI erweitert werden. Die relative URI hat die folgende Form:

```
/vN/Ergebnisse/AUFTRAGSID/Fehler/Fehler.txt       für Fehlerdokumente
/vN/Ergebnisse/AUFTRAGSID/Ausgabe/verbose.log     für Meldungsdokumente
/vN/Ergebnisse/AUFTRAGSID/Ausgabe/1               für Ausgabedokumente
```

Bei Ausgabedokumenten werden Ausgabe-Mappings geliefert (*siehe Beispielcode*[302]). Damit werden die einzelnen Ausgabedokument-URIs im Ergebnisdokument auf das entsprechende Dokument im Client Request gemappt.

Erweitern der relativen URI

Erweitern Sie die relative URI im Ergebnisdokument[301] zu einer absoluten HTTP-URI, indem Sie der relativen URI die Serveradresse voranstellen. Wenn die Serveradresse z.B. folgendermaßen lautet:

`http://localhost:8087/` (die Adresse in der Anfangskonfiguration[272])

und wenn die relative URI einer Fehlerdatei im Ergebnisdokument[301] folgendermaßen lautet:

```
/v1/results/20008201-219F-4790-BB59-C091C276FED2/error/error.txt
```

so ist die erweiterte absolute Adresse:

```
http://localhost:8087/v1/results/20008201-219F-4790-BB59-C091C276FED2/error/error.txt
```

Nähere Informationen im Zusammenhang damit finden Sie in den Abschnitten: Konfigurieren des Servers[272] und Abrufen des Ergebnisdokuments[301].

Verwenden eines HTTP `GET` Request zum Aufrufen der Datei

Verwenden Sie die erweiterte URI in einem HTTP `GET` Request, um die gewünschte Datei zu erhalten. RaptorXML Server gibt daraufhin das angeforderte Dokument zurück.

6.1.2.5 Freigeben von Serverressourcen nach der Verarbeitung

RaptorXML Server speichert die Ergebnisdokumentdatei, temporäre Dateien und Fehler- und Ausgabedokumente zu einem verarbeiteten Auftrag auf der Festplatte. Diese Dateien können auf zwei Arten gelöscht werden:

- Durch Angeben der URI des Ergebnisdokuments[301] mit der HTTP `DELETE`-Methode. Dadurch werden alle Dateien im Zusammenhang mit dem Auftrag, der durch die gesendete Ergebnisdokument-URI referenziert wird, einschließlich aller Fehler- und Ausgabedokumente gelöscht.
- Durch manuelles Löschen einzelner Dateien auf dem Server durch einen Administrator.

Die Struktur der URI, die mit der HTTP `DELETE`-Methode verwendet wird, entspricht der unten gezeigten. Beachten Sie, dass die vollständige URI aus der Serveradresse plus der relativen URI des Ergebnisdokuments besteht.

HTTP-Methode	URI
`DELETE`	`http://localhost:8087/v1/result/D405A84A-AB96-482A-96E7-4399885FAB0F`

Zur Angabe des Ausgabeverzeichnisses eines Auftrags auf der Festplatte, konstruieren Sie die URI folgendermaßen:

```
[<server.output-root-dir> see server configuration file[273]] + [jobid[301]]
```

Anmerkung: Da oft zahlreiche Fehler- und Ausgabedokumente erstellt werden, empfiehlt es sich, die Verwendung des Festplattenspeichers zu überwachen und je nach Umgebung und Ihren Anforderungen regelmäßig Löschungen vorzunehmen.

6.1.2.6 C#-Beispiel für die REST API

Ihre RaptorXML Server-Installation enthält ein C#-Projekt, das zur Ausführung einer Gruppe von Aufträgen die REST-Client-Schnittstelle von RaptorXML Server aufruft. Dieses Beispielprojekt besteht aus zwei Teilen:

- **`RaptorXMLREST.cs`**: einer Wrapper-Klasse in C#, die den REST-Mechanismus implementiert, um über HTTP mit RaptorXML Server zu kommunizieren.

- **Program.cs**: dem C#-Programmcode, der den Auftrag, der über den REST Wrapper an RaptorXML Server gesendet werden soll, definiert.

Diese beiden Teile werden in den folgenden Unterabschnitten beschrieben: C# Wrapper für die REST API 307 und Programmcode für REST Requests 308.

Beachten Sie, dass Sie für C#-Code jeden geeigneten REST Wrapper verwenden können. Der Hauptgrund dafür, dass wir unseren eigenen Wrapper erstellt haben, ist, dass das C#-Programm mit der Wrapper-Klasse enger integriert werden kann, wodurch die REST-Schnittstelle von RaptorXML Server verständlicher wird.

Pfad und Verwendung des C#-Beispiels

Das Beispielprojekt befindet sich im Ordner `C:\Programme (x86)\Altova\RaptorXML Server2025\examples\REST_API\C#_RaptorREST_API`.

Das Beispielprojekt wurde mit Visual Studio 2019 erstellt, daher sollten Sie zum Erstellen und Ausführen des Projekts diese oder eine höhere Version von Visual Studio verwenden. Beachten Sie, dass sich die C#-Beispieldateien im Ordner "Programme" befinden. Daher müssen Sie Visual Studio mit Administratorrechten öffnen, um Zugriff auf die Dateien zu erhalten. Alternativ dazu können Sie das Beispielprojekt in einen anderen Ordner kopieren und das Projekt entsprechend anpassen.

6.1.2.6.1 C# Wrapper für die REST API

Die Wrapper-Klasse ist in der C#-Datei mit dem Namen **RaptorXMLREST.cs** definiert und heißt **RaptorXMLRESTAPI**.

Sie definiert die folgenden Schlüsselklassen für das Senden von HTTP-Requests und das Empfangen der HTTP-Responses über REST:

- `Command`
- `MultiPartCommand`
- `CommandResponse`
- `ResultDocument`

Sie definiert die folgenden Funktionen:

- `pollCommandResult`
- `fetchCommandResult`
- `sendRequest`
- `cleanupResults`

Um zu sehen, wie die REST API im Wrapper implementiert wird, lesen Sie den Abschnitt Client Requests 282, um zu verstehen, wie die REST API funktioniert. Lesen Sie danach den C#-Code der Wrapper-Klasse, um zu sehen, wie der C#-Code für die REST API im Wrapper implementiert ist.

Wenn Sie z.B. sehen möchten, wie ein Befehl von C#-Code aus an RaptorXML Server gesendet wird, könnten Sie folgendermaßen vorgehen:

- Über die REST-Schnittstelle kann über einen HTTP **POST** Request ein Befehl an RaptorXML Server gesendet werden. Eine Beschreibung dazu finden Sie im Kapitel Initiieren von Aufträgen mittels POST[284].
- Die nächste Frage ist: Wie würde der Wrapper den Befehl an die REST API übergeben? Der Mechanismus dafür wird in der **Command**-Klasse des Wrappers implementiert. Öffnen Sie die Datei **RaptorXMLREST.cs**, um den Code der **Command**-Klasse zu sehen.
- Um schließlich zu sehen, wie die **Command**-Klasse des Wrappers im Programmcode[308] instantiiert wird, werfen Sie einen Blick in den Code der drei Aufträge im Programmcode[308].

6.1.2.6.2 Programmcode für REST Requests

Der C#-Programmcode, der die Aufträge fürRaptorXML Server enthält, ist in der C#-Datei namens **Program.cs** definiert. Im Code werden die im C# Wrapper für die REST API[307] definierten Klassen verwendet, um die an RaptorXML Server gesendeten Requests zu erstellen.

Der Programmcode enthält drei Anwendungsbeispiele für die Verwendung der REST API von RaptorXML Server:

- Validierung einer referenzierten XML-Datei[308] mit Hilfe des **valany**[221]-Befehls von RaptorXML Server. Die Schema-Datei wird von der XML-Datei aus referenziert und muss nicht als Argument des Befehls angegeben werden.
- Zwei XML-Dateien werden mit Hilfe des **valany**[221]-Befehls von RaptorXML Server validiert[309]. Beide XML-Dateien sowie die für die Validierung verwendete Schema-Datei werden mit dem Befehl als String-Anhänge hochgeladen. Die Ergebnisse der Validierung werden nach Abschluss beider Validierungen gemeinsam zurückgegeben.
- Eine XML-Datei wird hochgeladen und mit Hilfe einer XSLT-Datei transformiert[309]. Beide Dateien werden über REST hochgeladen. Dazu wird der Befehl **xslt**[130] von RaptorXML Server verwendet. Das mit der Transformation erzeugte Dokument wird durch das Programm aufgerufen.

Weiter unten finden Sie eine nähere Erklärung zum Code dieser drei Anwendungsbeispiele.

Fehlerbehandlung

Im Fall eines Fehlers wird die Fehlermeldung mit Hilfe einer Fehlerbehandlungsfunktion (namens **HandleError**) im unteren Teil des Codes aus der Server Response[298] abgerufen.

Fall 1: Validierung einer referenzierten XML-Datei (einfacher Befehl)

Im Programmcode für dieses Anwendungsbeispiel werden Klassen und Funktion aus dem REST API Wrapper verwendet, um die HTTP-Kommunikation mit RaptorXML Server einzurichten und auszuführen. Die Logik des Codes ist die folgende:

RaptorXMLRESTAPI.Command	Definiert den aufzurufenden RaptorXML Server-Befehl, nämlich **valany**, und die Datei, die als das Argument des `valany`Befehls bereitgestellt werden soll.
RaptorXMLRESTAPI.CommandResponse	Setzt die Antwort des Servers auf den Validierungs-Request in die Variable **jsonResponse**. Beachten Sie, dass Validierungsaufträge das Resultat "OK" oder "Fehlgeschlagen"[300] haben.

RaptorXMLRESTAPI.ResultDocument	Ruft das vom Server zurückgegebene Ergebnisdokument auf und zeigt das Validierungsergebnis an, falls keine Fehler aufgetreten sind.

Fall 2: Validierung zweier hochgeladener XML-Dateien anhand einer hochgeladenen XSD-Datei (multipart-Befehl)

Im Programmcode für dieses Anwendungsbeispiel wird die Klasse **MultiPartCommand** aus dem REST API Wrapper verwendet, um die HTTP-Kommunikation mit RaptorXML Server einzurichten und auszuführen. Da Dateien im Body des POST Request hochgeladen werden sollen, muss der Inhaltstyp des Message Headers auf multipart/form-data[284] gesetzt werden. Mit Hilfe der Klasse **MultiPartCommand** des Wrappers wird die REST HTTP-Kommunikation entsprechend konfiguriert. Der Code für dieses Anwendungsbeispiel ist folgendermaßen strukturiert:

RaptorXMLRESTAPI.MultiPartCommand	Definiert den aufzurufenden RaptorXML Server-Befehl, nämlich **valany**, und lädt die beiden XML-Dateien und die Schema-Datei anschließend mit Hilfe der Funktion **AppendAttachment** der Klasse hoch. Die Dateien werden als Strings bereitgestellt. Die Server Response gibt das Validierungsergebnis der beiden Dateien zurück und diese Antwort wird in der Variablen **jsonResponse** gespeichert.
RaptorXMLRESTAPI.fetchCommandResult	Ruft das vom Server zurückgegebene Ergebnisdokument auf und zeigt die Validierungsergebnisse an, falls keine Fehler aufgetreten sind.
RaptorXMLRESTAPI.cleanupResults	Diese Funktion des Wrappers löscht die Ergebnisdokumentdatei, die temporären Dateien und die mit dem Auftrag in Zusammenhang stehenden Fehler- und Ausgabedokumentdateien mit Hilfe der **DELETE**-Methode von HTTP.

Fall 3: XSLT-Transformation hochgeladener XML- und XSLT-Dateien (multipart-Befehl)

Der Programmcode für dieses Beispiel ähnelt dem des Falls 2 weiter oben. Darin wird die Klasse **MultiPartCommand** verwendet, um eine XSLT-Transformation einzurichten und das Ergebnisdokument in einem Meldungsfeld anzuzeigen. Die XML- und XSLT-Datei für die Transformation werden mit dem Request hochgeladen. Der **XSLT**-Befehl von RaptorXML Server kann außerdem Optionen haben, daher wird in diesem Beispiel gezeigt, wie Sie Optionen über die REST-Schnittstelle hinzufügen können (in diesem Beispiel erfolgt dies mit Hilfe der **RaptorXMLRESTAPI.AppendOption**-Funktion. Im Folgenden finden Sie wichtige Anmerkungen zum Code.

RaptorXMLRESTAPI.MultiPartCommand	Definiert den aufzurufenden RaptorXML Server-Befehl **XSLT** und verwendet anschließend (i) die Funktion **AppendAttachment** der Klasse, um die XML- und XSLT-Datei hochzuladen und (ii) die Funktion **AppendOption**, um Optionen für die RaptorXML Server-Befehlszeile bereitzustellen. Die hochgeladenen Dateien werden als Strings bereitgestellt. Die Server Response gibt das Validierungsergebnis der beiden Dateien zurück und diese Antwort wird in der Variablen **jsonResponse** gespeichert.

RaptorXMLRESTAPI.fetchCommandResult	Ruft das vom Server zurückgegebene Ergebnisdokument auf und zeigt die Validierungsergebnisse an, falls keine Fehler aufgetreten sind.
RaptorXMLRESTAPI.cleanupResults	Diese Funktion des Wrappers löscht die Ergebnisdokumentdatei, die temporären Dateien und die mit dem Auftrag in Zusammenhang stehenden Fehler- und Ausgabedokumentdateien mit Hilfe der **DELETE**-Methode von HTTP.

6.2 COM/.NET API

RaptorXML Server ist auf dem Rechner, auf dem er installiert ist, lizenziert. Die .NET-Schnittstelle bildet einen Wrapper rund um die COM-Schnittstelle. Die COM- und die .NET-Schnittstelle von RaptorXML Server verwenden eine einzige API: die COM/.NET API von RaptorXML Server (Objektreferenz hier 323).

Sie können RaptorXML mit folgenden Sprachen verwenden:

- Skriptsprachen wie JavaScript, über die COM-Schnittstelle
- Programmiersprachen wie C#, über das .NET Framework

6.2.1 COM-Schnittstelle

RaptorXML Server wird bei der Installation von RaptorXML Server automatisch als COM-Serverobjekt registriert, sodass das Programm von Applikationen und Skriptsprachen mit Unterstützung für COM-Aufrufe aufgerufen werden kann. Wenn Sie den Pfad des RaptorXML Server Installationspakets ändern möchten, sollten Sie RaptorXML Server am besten zuerst deinstallieren und anschließend im gewünschten Ordner installieren. Auf diese Art wird die Aufhebung der Registrierung und die erneute Registrierung automatisch während der Installation vorgenommen.

Überprüfen, ob die Registrierung erfolgreich vorgenommen wurde

Wenn die Registrierung erfolgreich war, enthält die Registrierungsdatei die Klassen `RaptorXML.Server` Klassen. Diese Klassen befinden sich normalerweise unter `HKEY_LOCAL_MACHINE\SOFTWARE\Classes`.

Codebeispiel

- Im folgenden Kapitel finden Sie ein VBScript-Beispiel 311, in dem gezeigt wird, wie die RaptorXML API über ihre COM-Schnittstelle verwendet werden kann.
- Eine Beispieldatei dazu finden Sie im RaptorXML Applikationsordner im Ordner `examples/API`.

6.2.2 COM-Beispiel: VBScript

Das VBScript-Beispiel ist in die folgenden Abschnitte gegliedert:


```
' The RaptorXML COM object
dim objRaptor

' Initialize the RaptorXML COM object
sub Init
	objRaptor = Null
```

```
	On Error Resume Next
	' Try to load the 32-bit COM object; do not throw exceptions if object is not found
	Set objRaptor = WScript.GetObject( "", "RaptorXML.Server" )
	On Error Goto 0
	if ( IsNull( objRaptor ) ) then
		' Try to load the 64-bit object (exception will be thrown if not found)
		Set objRaptor = WScript.GetObject( "", "RaptorXML_x64.Server" )
	end if
	' Configure the server: error reporting, HTTP server name and port (IPv6 localhost
in this example)
	objRaptor.ErrorLimit = 1
	objRaptor.ReportOptionalWarnings = true
	objRaptor.ServerName = "::1"
	objRaptor.ServerPort = 8087
end sub


' Validate one file
sub ValidateXML
	' Get a validator instance from the Server object
	dim objXMLValidator
	Set objXMLValidator = objRaptor.GetXMLValidator()

	' Configure input data
	objXMLValidator.InputFileName = "MyXMLFile.xml"

	' Validate; in case of invalid file report the problem returned by RaptorXML
	if ( objXMLValidator.IsValid() ) then
		MsgBox( "Input string is valid" )
	else
		MsgBox( objXMLValidator.LastErrorMessage )
	end if
end sub


' Perform a transformation; return the result as a string
sub RunXSLT
	' Get an XSLT engine instance from the Server object
	dim objXSLT
	set objXSLT = objRaptor.GetXSLT

	' Configure input data
	objXSLT.InputXMLFileName = "MyXMLFile.xml"
	objXSLT.XSLFileName = "MyTransformation.xsl"

	' Run the transformation; in case of success the result will be returned, in case of
errors the engine returns an error listing
	MsgBox( objXSLT.ExecuteAndGetResultAsString() )
end sub


' Execute an XQuery; save the result in a file
sub RunXQuery
	' Get an XQuery engine instance from the Server object
```



```
        dim objXQ
        set objXQ = objRaptor.GetXQuery()

        ' Configure input data
        objXQ.InputXMLFileName = "MyXMLFile.xml"
        objXQ.XQueryFileName = "MyQuery.xq"

        ' Configure serialization (optional - for fine-tuning the result's formatting)
        objXQ.OutputEncoding = "UTF8"
        objXQ.OutputIndent = true
        objXQ.OutputMethod = "xml"
        objXQ.OutputOmitXMLDeclaration = false

        ' Run the query; the result will be serialized to the given path
        call objXQ.Execute( "MyQueryResult.xml" )
end sub


' Perform all sample functions
sub main
        Init
        ValidateXML
        RunXSLT
        RunXQuery
end sub

' Script entry point; run the main function
main
```

6.2.3 .NET-Schnittstelle

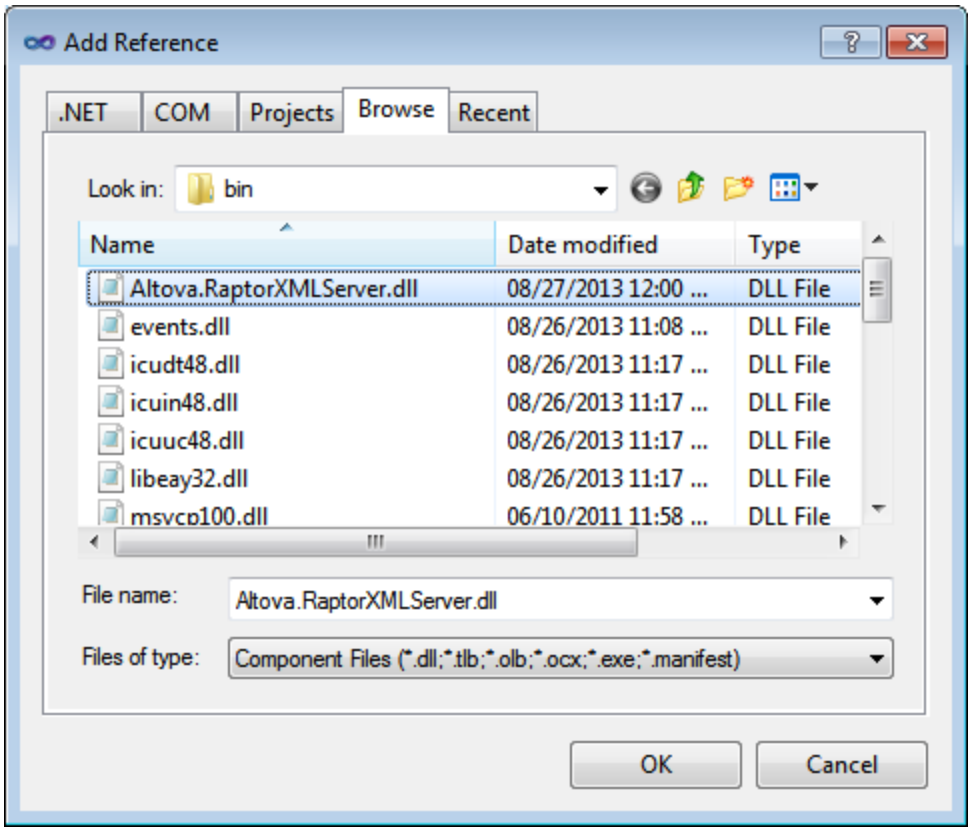
Die .NET-Schnittstelle ist als Wrapper rund um die RaptorXML COM-Schnittstelle gebaut. Sie dient als primäre von Altova signierte Interop-Assembly und verwendet den Namespace `Altova.RaptorXMLServer`.

Hinzufügen der RaptorXML DLL als Referenz zum Visual Studio .NET-Projekt

Um RaptorXML in Ihrem .NET-Projekt verwenden zu können, fügen Sie in Ihrem Projekt eine Referenz zur RaptorXML-DLL (`Altova.RaptorXMLServer.dll`) hinzu. Ihre RaptorXML Server Installation enthält eine signierte DLL-Datei namens `Altova.RaptorXMLServer.dll`, die bei der Installation von RaptorXML mit Hilfe des RaptorXML Installationsprogramms automatisch zum globalen Assembly Cache (GAC) hinzugefügt wird. Normalerweise befindet sich der GAC im Ordner `C:\WINDOWS\assembly`.

Um die RaptorXML-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



1. Navigieren Sie auf dem Register "Browse" zum Ordner: `<RaptorXML application folder>/bin`, wählen Sie die RaptorXML DLL `Altova.RaptorXMLServer.dll` aus und klicken Sie auf **OK**.
2. Wählen Sie den Befehl View | Object Browser, um die Objekte der RaptorXML API zu sehen.

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

Anmerkung: RaptorXML wird bei der Installation automatisch als COM-Serverobjekt registriert, daher ist eine manuelle Registrierung nicht erforderlich.

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

Codebeispiele

In den folgenden Kapiteln finden Sie ein C#-Beispiel 315 und ein Visual Basic .NET-Beispiel 317 zur Verwendung der RaptorXML API über ihre .NET-Schnittstelle. Die Dateien zu diesen Beispielen befinden sich im RaptorXML-Applikationsordner im Ordner `examples/serverAPI`.

6.2.4 .NET-Beispiel: C#

Das C#-Beispiel ist in die folgenden Abschnitte gegliedert:


```
using System;
using System.Text;
using Altova.RaptorXMLServer;

namespace RaptorXMLRunner
{
    class Program
    {
    // The RaptorXML Server .NET object
       static ServerClass objRaptorXMLServer;

    // Initialize the RaptorXML Server .NET object
       static void Init()
       {
       // Allocate a RaptorXML Server object
          objRaptorXMLServer = new ServerClass();

       // Configure the server: error reporting, HTTP server name and port
       // (IPv6 localhost in this example)
          objRaptorXMLServer.ErrorLimit = 1;
          objRaptorXMLServer.ReportOptionalWarnings = true;
          objRaptorXMLServer.ServerName = "::1"
          objRaptorXMLServer.ServerPort = 8087
       }



       // Validate one file
          static void ValidateXML()
          {
          // Get a validator engine instance from the Server object
             XMLValidator objXMLValidator = objRaptorXMLServer.GetXMLValidator();

          // Configure input data
             objXMLValidator.InputFileName = "MyXMLFile.xml";

          // Validate; in case of invalid file,
             report the problem returned by RaptorXML
             if ( objXMLValidator.IsValid() )
                Console.WriteLine( "Input string is valid" );
             else
```

```
            Console.WriteLine( objXMLValidator.LastErrorMessage );
      }



   // Perform an XSLT transformation, and
   // return the result as a string
      static void RunXSLT()
      {
      // Get an XSLT engine instance from the Server object
         XSLT objXSLT = objRaptorXMLServer.GetXSLT();

         // Configure input data
            objXSLT.InputXMLFileName = "MyXMLFile.xml";
            objXSLT.XSLFileName = "MyTransformation.xsl";

        // Run the transformation.
        // In case of success, the result is returned.
        // In case of errors, an error listing
           Console.WriteLine( objXSLT.ExecuteAndGetResultAsString() );
      }



   // Execute an XQuery, save the result in a file
      static void RunXQuery()
      {
      // Get an XQuery engine instance from the Server object
         XQuery objXQuery = objRaptorXMLServer.GetXQuery();

        // Configure input data
           objXQuery.InputXMLFileName = exampleFolder + "simple.xml";
           objXQuery.XQueryFileName = exampleFolder + "CopyInput.xq";

       // Configure serialization (optional, for better formatting)
          objXQuery.OutputEncoding = "UTF8"
          objXQuery.OutputIndent = true
          objXQuery.OutputMethod = "xml"
          objXQuery.OutputOmitXMLDeclaration = false

       // Run the query; result serialized to given path
          objXQuery.Execute( "MyQueryResult.xml" );
      }



      static void Main(string[] args)
      {
        try
        {
        // Entry point. Perform all functions
           Init();
```



```
                ValidateXML();
                RunXSLT();
                RunXQuery();
            }
            catch (System.Exception ex)
            {
                Console.WriteLine( ex.Message );
                Console.WriteLine( ex.ToString() );
            }
        }
    }
}
```

6.2.5 .NET-Beispiel: Visual Basic .NET

Das VBScript-Beispiel ist in die folgenden Abschnitte gegliedert:

- Einrichten und Initialisieren des RaptorXML .NET-Objekts 317
- Validieren einer XML-Datei 317
- Durchführung einer XSLT-Transformation, Rückgabe des Ergebnisses als String 318
- Verarbeiten eines XQuery-Dokuments, Speichern des Ergebnisses in einer Datei 318
- Einrichten der Ausführungssequenz des Skripts und seines Eintrittspunkts 318

```
Option Explicit On
Imports Altova.RaptorXMLServer

Module RaptorXMLRunner

   ' The RaptorXML .NET object
     Dim objRaptor As Server

   ' Initialize the RaptorXML .NET object
     Sub Init()

      ' Allocate a RaptorXML object
        objRaptor = New Server()

      ' Configure the server: error reporting, HTTP server name and port (IPv6 localhost in
this example)
        objRaptor.ErrorLimit = 1
        objRaptor.ReportOptionalWarnings = True
        objRaptor.ServerName = "::1"
        objRaptor.ServerPort = 8087
     End Sub


   ' Validate one file
     Sub ValidateXML()

      ' Get a validator instance from the RaptorXML object
        Dim objXMLValidator As XMLValidator
        objXMLValidator = objRaptor.GetXMLValidator()
```

```
      ' Configure input data
        objXMLValidator.InputFileName = "MyXMLFile.xml"

      ' Validate; in case of invalid file report the problem returned by RaptorXML
        If (objXMLValidator.IsValid()) Then
           Console.WriteLine("Input string is valid")
        Else
           Console.WriteLine(objXMLValidator.LastErrorMessage)
        End If
    End Sub


  ' Perform a transformation; return the result as a string
    Sub RunXSLT()

      ' Get an XSLT engine instance from the Server object
        Dim objXSLT As XSLT
        objXSLT = objRaptor.GetXSLT()

      ' Configure input data
        objXSLT.InputXMLFileName = "MyXMLFile.xml"
        objXSLT.XSLFileName = "MyTransformation.xsl"

      ' Run the transformation; in case of success the result will be returned, in case of
errors the engine returns an error listing
        Console.WriteLine(objXSLT.ExecuteAndGetResultAsString())
    End Sub


  ' Execute an XQuery; save the result in a file
    Sub RunXQuery()

      ' Get an XQuery engine instance from the Server object
        Dim objXQ As XQuery
        objXQ = objRaptor.GetXQuery()

      ' Configure input data
        objXQ.InputXMLFileName = "MyXMLFile.xml"
        objXQ.XQueryFileName = "MyQuery.xq"

      ' Configure serialization (optional - for fine-tuning the result's formatting)
        objXQ.OutputEncoding = "UTF8"
        objXQ.OutputIndent = true
        objXQ.OutputMethod = "xml"
        objXQ.OutputOmitXMLDeclaration = false

      ' Run the query; the result will be serialized to the given path
        objXQ.Execute( "MyQueryResult.xml" )
    End Sub


    Sub Main()
      ' Entry point; perform all sample functions
        Init()
        ValidateXML()
```



```
        RunXSLT()
        RunXQuery()
    End Sub

End Module
```

6.3 Java API

Die RaptorXML API kann von Java-Code aus aufgerufen werden. Dazu müssen die unten aufgelisteten Bibliotheken im Classpath aufgelistet sein. Diese Bibliotheken sind im Installationsordner im `bin`-Ordner installiert.

- `RaptorXMLServer.jar`: Die Bibliothek, die über HTTP-Requests mit dem RaptorXML Server kommuniziert.
- `RaptorXMLServer_JavaDoc.zip`: Eine Javadoc-Datei, die die Hilfedokumentation zur Java API enthält.

Note: Um die Java API verwenden zu können, muss sich die Jar-Datei im Java-Classpath befinden. Sie können die Jar-Datei in jeden beliebigen Ordner kopieren, falls dies für Ihre Projektkonfiguration besser geeignet ist als das Referenzieren der Datei vom ursprünglich installierten Pfad aus.

6.3.1 Überblick über die Schnittstelle

Die Java-API ist im `com.altova.raptorxml`-Paket verpackt. Die `RaptorXML`-Klasse stellt als Eintrittspunkt eine Methode namens `getFactory()` zur Verfügung, die `RaptorXMLFactory`[323]-Objekte bereitstellt. Dadurch kann mit dem Aufruf: `RaptorXML.getFactory()` eine `RaptorXMLFactory`-Instanz erstellt werden.

Die `RaptorXMLFactory`[323]-Schnittstelle enthält Methoden zum Aufrufen von Prozessorobjekten für die Validierung und andere Verarbeitungsfunktionalitäten (wie z.B. die XSLT-Transformation).

RaptorXMLFactory

Die öffentliche Schnittstelle `RaptorXMLFactory`[323] wird durch das folgende Codefragment beschrieben:

```
public interface RaptorXMLFactory
{
   public XMLValidator[340] getXMLValidator();
   public XMLDSig[333] getXMLDSig();
   public XQuery[353] getXQuery();
   public XSLT[365] getXSLT();
   public void setServerName(String name) throws RaptorXMLException[333];
   public void setServerPath(String path) throws RaptorXMLException[333];
   public void setServerPort(int port) throws RaptorXMLException[333];
   public void setGlobalCatalog(String catalog);
   public void setUserCatalog(String catalog);
   public void setGlobalResourcesFile(String file);
   public void setGlobalResourceConfig(String config);
   public void setErrorFormat(RaptorXMLException[333] format);
   public void setErrorLimit(int limit);
   public void setReportOptionalWarnings(boolean report);
}
```

Nähere Informationen dazu finden Sie unter der Beschreibung zu `RaptorXMLFactory`[323] und den dazugehörigen Java-Methoden. Siehe auch Java-Beispielprojekt[321].

6.3.2 Java-Beispielprojekt

Im nachstehenden Java-Codefragment wird gezeigt, wie grundlegende Funktionalitäten aufgerufen werden können. Der Abschnitt ist in die folgenden Unterabschnitte gegliedert:

Diese Grundfunktionalität ist in den Dateien im RaptorXML Server-Applikationsordner im Ordner `examples/API` enthalten.

```
public class RunRaptorXML
{
    // Locate samples installed with the product
    // (will be two levels higher from examples/API/Java)
    // REMARK: You might need to modify this path
       static final String strExamplesFolder = System.getProperty("user.dir") + "/../../" ;

       static com.altova.raptorxml.RaptorXMLFactory rxml;



       static void ValidateXML() throws com.altova.raptorxml.RaptorXMLException
       {
           com.altova.raptorxml.XMLValidator xmlValidator = rxml.getXMLValidator();
           System.out.println("RaptorXML Java - XML validation");
           xmlValidator.setInputFromText( "<!DOCTYPE root [ <!ELEMENT root (#PCDATA)> ]>
<root>simple input document</root>" );
              if( xmlValidator.isWellFormed() )
                  System.out.println( "The input string is well-formed" );
              else
                  System.out.println( "Input string is not well-formed: " +
xmlValidator.getLastErrorMessage() );

              if( xmlValidator.isValid() )
                  System.out.println( "The input string is valid" );
              else
                  System.out.println( "Input string is not valid: " +
xmlValidator.getLastErrorMessage() );
       }



       static void RunXSLT() throws com.altova.raptorxml.RaptorXMLException
       {
           System.out.println("RaptorXML Java - XSL Transformation");
           com.altova.raptorxml.XSLT xsltEngine = rxml.getXSLT();
           xsltEngine.setInputXMLFileName( strExamplesFolder + "simple.xml" );
           xsltEngine.setXSLFileName( strExamplesFolder + "transform.xsl" );
```

```
            String result = xsltEngine.executeAndGetResultAsString();
            if( result == null )
                System.out.println( "Transformation failed: " +
xsltEngine.getLastErrorMessage() );
            else
                System.out.println( "Result is " + result );
        }



        static void RunXQuery() throws com.altova.raptorxml.RaptorXMLException
        {
            System.out.println("RaptorXML Java - XQuery execution");
            com.altova.raptorxml.XQuery xqEngine = rxml.getXQuery();
            xqEngine.setInputXMLFileName( strExamplesFolder + "simple.xml" );
            xqEngine.setXQueryFileName( strExamplesFolder + "CopyInput.xq" );
            System result = xqEngine.executeAndGetResultAsString();
            if( result == null )
                System.out.println( "Execution failed: " + xqEngine.getLastErrorMessage() );
            else
                System.out.println( "Result is " + result );
        }



        public static void main(String[] args)
        {
            try
            {
                rxml = com.altova.raptorxml.RaptorXML.getFactory();
                rxml.setErrorLimit( 3 );

                ValidateXML();
                RunXSLT();
                RunXQuery();
            }

            catch( com.altova.raptorxml.RaptorXMLException e )
            {
                e.printStackTrace();
            }

        }

}
```

6.4 Referenz zur Server API

In diesem Abschnitt wird die RaptorXML Server API beschrieben: ihr Objektmodell und die Einzelheiten ihrer Schnittstellen und Enumerationen. Die Beschreibung der API gilt sowohl für die COM/.NET- als auch für die Java-Schnittstelle. Zwar ist die Struktur der API für beide Schnittstellen dieselbe, doch unterscheiden sich die Namen von Methoden und Eigenschaften. Aus diesem Grund sind die einzelnen Methoden, Eigenschaften und Enumerationen mit einer separaten Signatur für COM/.NET und Java beschrieben.

Der Ausgangspunkt für die Verwendung der Funktionalitäten von RaptorXML Server ist die `IServer`[323]-Schnittstelle (COM/.NET) bzw. die `RaptorXMLFactory`[323]-Klasse (Java).

6.4.1 Schnittstellen/Klassen

Der Ausgangspunkt für die Verwendung der Funktionalitäten von RaptorXML ist die `IServer`[323]-Schnittstelle (COM/.NET) bzw. die `RaptorXMLFactory`[323]-Klasse (Java). Dieses Objekt enthält die Objekte, die die RaptorXML-Funktionalitäten bereitstellen: XML-Validierung, Verarbeitung von XQuery-Dokumenten und XML-Signaturen und XSLT-Transformationen.

Unten sehen Sie die Hierarchie des Objektmodells. Die Schnittstellen werden in den entsprechenden Abschnitten näher beschrieben. Die Methoden und Eigenschaften der einzelnen Schnittstellen sind im Abschnitt zur jeweiligen Schnittstelle beschrieben.

IServer *(COM/.NET)* **/ RaptorXMLFactory** *(Java)*
  |-- **IXMLDSig***(COM/.NET)* **/ XMLDSig** *(Java)*
  |-- **IXMLValidator** *(COM/.NET)* **/ XMLValidator** *(Java)*
  |-- **IXSLT** *(COM/.NET)* **/ XSLT** *(Java)*
  |-- **IXQuery** *(COM/.NET)* **/ XQuery** *(Java)*

6.4.1.1 IServer/RaptorXMLFactory

Über die **`IServer/RaptorXMLFactory`**-Schnittstelle können Sie den gewünschten RaptorXML-Prozessor aufrufen. Beachten Sie, dass die Schnittstellen in der COM/.NET API einen anderen Namen haben als die in der Java API:

- In COM/.NET: **`IServer`**
- In Java: **`RaptorXMLFactory`**

In diesem Abschnitt werden die Methoden und Eigenschaften von **`IServer/RaptorXMLFactory`** beschrieben.

Java API-Eintrittspunktmethode

Die Java-API ist im `com.altova.raptorxml`-Paket verpackt. Die `RaptorXML`-Klasse stellt als Eintrittspunkt eine Methode namens `getFactory()` zur Verfügung, die `RaptorXMLFactory`[323]-Objekte bereitstellt. Dadurch kann mit dem Aufruf: `RaptorXML.getFactory()` eine `RaptorXMLFactory`-Instanz erstellt werden.

6.4.1.1.1 Methoden

Die Methoden der **`IServer`** (COM/.NET) und **`RaptorXMLFactory`** (Java)-Schnittstelle geben eine Instanz des entsprechenden RaptorXML -Prozessors bzw. der entsprechenden RaptorXML-Klasse zurück: XMLDSig, XML Validator, XSLT und XQuery.

COM/.NET	Java
GetXMLDsig[324] *(für XML-Signaturen)*	getXMLDsig[324] *(für XML-Signaturen)*
GetXMLValidator[324]	getXMLValidator[324]
GetXQuery[325]	getXQuery[325]
GetXSLT[325]	getXSLT[325]

6.4.1.1.1.1 GetXMLDsig (für XML-Signaturen)

Gibt eine Instanz der XML-Signatur-Schnittstelle/Klasse zurück (**`XMLDSig`**[333]).

COM und .NET

Signatur: **`IXMLDSig`**[333] **`GetXMLDSig()`**

Java

Signatur: **`public XMLDSig`**[333] **`getXMLDSig()`**

6.4.1.1.1.2 GetXMLValidator

Gibt eine Instanz des XML-Validierungsprozessors zurück.

COM und .NET

Signatur: **`IXMLValidator`**[340] **`GetXMLValidator()`**

Java

Signatur: **`public XMLValidator`**[340] **`getXMLValidator()`**

6.4.1.1.1.3 GetXQuery

Gibt eine Instanz des XQuery-Prozessors zurück.

COM und .NET

Signatur: **IXQuery**[353] **GetXQuery()**

Java

Signatur: **public XQuery**[353] **getXQuery()**

6.4.1.1.1.4 GetXSLT

Gibt eine Instanz des XSLT-Prozessors zurück.

COM und .NET

Signatur: **IXSLT**[365] **GetXSLT()**

Java

Signatur: **public XSLT**[365] **getXSLT()**

6.4.1.1.2 Eigenschaften

In diesem Abschnitt sind die Eigenschaften der **IServer** (COM/.NET) und **RaptorXMLFactory** (Java)-Schnittstelle beschrieben.

COM/.NET	Java
APIMajorVersion[326]	getAPIMajorVersion[326]
APIMinorVersion[326]	getAPIMinorVersion[326]
APIServicePackVersion[327]	getAPIServicePackVersion[327]
ErrorFormat[327]	setErrorFormat[327]
ErrorLimit[327]	setErrorLimit[327]
GlobalCatalog[328]	setGlobalCatalog[328]

GlobalResourceConfig 328	setGlobalResourceConfig 328
GlobalResourcesFile 328	setGlobalResourcesFile 328
Is64Bit 329	ss64Bit 329
MajorVersion 329	getMajorVersion 329
MinorVersion 329	getMinorVersion 329
ProductName 330	getProductName 330
ProductNameAndVersion 330	getProductNameAndVersion 330
ReportOptionalWarnings 330	setReportOptionalWarnings 330
ServerName 331	setServerName 331
ServerPath 331	setServerPath 331
ServerPort 331	setServerPort 331
ServicePackVersion 332	getServicePackVersion 332
UserCatalog 332	setUserCatalog 332

6.4.1.1.2.1 APIMajorVersion

Gibt die Hauptversion der API als Ganzzahl zurück. Die API-Hauptversion muss nicht unbedingt mit der Hauptversion des Produkts 329 übereinstimmen, wenn die API mit einem anderen Server verbunden ist.

COM und .NET

Signatur: `int APIMajorVersion()`

Java

Signatur: `public int getAPIMajorVersion()`

6.4.1.1.2.2 APIMinorVersion

Gibt die Nebenversion der API als Ganzzahl zurück. Die API-Nebenversion muss nicht unbedingt mit der Nebenversion des Produkts 329 übereinstimmen, wenn die API mit einem anderen Server verbunden ist.

COM und .NET

Signatur: `int APIMinorVersion()`

Java

Signatur: `public int getAPIMinorVersion()`

6.4.1.1.2.3 APIServicePackVersion

Gibt die Service Pack-Version der API als Ganzzahl zurück. Die Service Pack-Version der API muss nicht unbedingt mit der Service Pack-Version des Produkts[332] übereinstimmen, wenn die API mit einem anderen Server verbunden ist.

COM und .NET

Signatur: `int APIServicePackVersion()`

Java

Signatur: `public int getAPIServicePackVersion()`

6.4.1.1.2.4 ErrorFormat

Definiert als das RaptorXML-Fehlerformat eines der `ENUMErrorFormat`[376]-Literale (Text, ShortXML, LongXML).

COM und .NET

Signatur: `ErrorFormat(ENUMErrorFormat[376] format)`

Java

Signatur: `public void setErrorFormat(ENUMErrorFormat[376] format)`

6.4.1.1.2.5 ErrorLimit

Definiert das Limit für RaptorXML-Validierungsfehler. Der Parameter `limit` hat den Typ `int` (Java), `uint` (COM/.NET) und definiert die Anzahl der Fehler, die ausgegeben werden, bevor die Ausführung gestoppt wird. Mit Hilfe von `-1` können Sie `limit` auf unbegrenzt setzen (d.h. alle Fehler werden ausgegeben). Der Standardwert ist `100`.

COM und .NET

Signatur: `ErrorLimit(uint limit)`

Java

Signatur: `public int setErrorLimit(int limit)`

6.4.1.1.2.6 GlobalCatalog

Definiert den Pfad der Hauptkatalogdatei (Eintrittspunkt) in Form einer URL. Der bereitgestellte String muss eine absolute URL sein, die den genauen Pfad zur Hauptkatalogdatei, die verwendet werden soll, angibt.

COM und .NET

Signatur: `GlobalCatalog(string catalog)`

Java

Signatur: `public void setGlobalCatalog(string catalog)`

6.4.1.1.2.7 GlobalResourceConfig

Definiert die aktive Konfiguration der globalen Ressource. Der Parameter `config` ist vom Typ `String` und gibt den Namen der von der aktiven globalen Ressource zu verwendenden Konfiguration an.

COM und .NET

Signatur: `GlobalResourceConfig(string config)`

Java

Signatur: `public void setGlobalResourceConfig(string config)`

6.4.1.1.2.8 GlobalResourcesFile

Definiert den Pfad zur XML-Datei für globale Ressourcen als URL. Beim bereitgestellten String muss es sich um eine absolute URL handeln, die den genauen Pfad zur XML-Datei für globale Ressourcen angibt.

COM und .NET

Signatur: `GlobalResourcesFile(string url)`

Java

Signatur: `public void setGlobalResourcesFile(string url)`

6.4.1.1.2.9 Is64Bit

Überprüft, ob die Applikation eine ausführbare 64-Bit-Datei ist. Gibt den Booleschen Wert `true` zurück, wenn die Applikation eine 64-Bit-Applikation ist, andernfalls `false`. *Beispiel:* Gibt für `Altova RaptorXML Server 2025r2sp1(x64)` den Wert `true` zurück. Löst bei Fehler eine **RaptorXMLException** 333 aus.

COM und .NET

Signatur: `boolean Is64Bit()`

Java

Signatur: `public boolean is64Bit()`

6.4.1.1.2.10 MajorVersion

Gibt die Hauptversion des Produkts als Ganzzahl zurück. *Beispiel:* Gibt für `Altova RaptorXML Server 2025 2018r2sp1(x64)` den Wert `20` (Unterschied zwischen der Hauptversion (`2018`) und dem Anfangsjahr `1998`) zurück. Löst bei Fehler eine **RaptorXMLException** 333 aus.

COM und .NET

Signatur: `int MajorVersion()`

Java

Signatur: `public int getMajorVersion()`

6.4.1.1.2.11 MinorVersion

Gibt die Nebenversion des Produkts als Ganzzahl zurück. *Beispiel:* Gibt für `Altova RaptorXML Server 2025r2sp1(x64)` den Wert `2` (aus der Nebenversionsnummer `r2`) zurück. Löst bei Fehler eine **RaptorXMLException** 333 aus.

COM und .NET

Signatur: `int` **`MinorVersion()`**

Java

Signatur: `public int` **`getMinorVersion()`**

6.4.1.1.2.12 ProductName

Gibt den Namen des Produkts als String zurück. *Beispiel:* Gibt für `Altova RaptorXML Server 2025r2sp1(x64)` den Wert `Altova RaptorXML Server` zurück. Löst bei Fehler eine **`RaptorXMLException`** 333 aus.

COM und .NET

Signatur: `string` **`ProductName()`**

Java

Signatur: `public string` **`getProductName()`**

6.4.1.1.2.13 ProductNameAndVersion

Gibt den Produktnamen, die Hauptversion, Nebenversion und Service Pack-Version des Produkts als String zurück. *Beispiel:* Gibt für `Altova RaptorXML Server 2025r2sp1(x64)` den Wert `Altova RaptorXML Server 2025r2sp1(x64)` zurück. Löst bei Fehler eine **`RaptorXMLException`** 333 aus.

COM und .NET

Signatur: `string` **`ProductNameAndVersion()`**

Java

Signatur: `public string` **`getProductNameAndVersion()`**

6.4.1.1.2.14 ReportOptionalWarnings

Aktiviert/Deaktiviert die Ausgabe von Warnungen. Mit dem Wert `true` werden Warnungen aktiviert; mit `false` werden sie deaktiviert.

COM und .NET

Signatur: `ReportOptionalWarnings(boolean report)`

Java

Signatur: `public void setReportOptionalWarnings(boolean report)`

6.4.1.1.2.15 ServerName

Definiert den Namen des HTTP-Servers, über den eine Verbindung zu RaptorXML Server hergestellt wird. Der Input-Parameter ist ein String, der den Namen des HTTP-Servers angibt. Verursacht eine **RaptorXMLException** 333, wenn ein Fehler auftritt.

COM und .NET

Signatur: `ServerName(string name)`

Java

Signatur: `public void setServerName(string name)`

6.4.1.1.2.16 ServerPath

Definiert den Pfad zum HTTP-Server in Form einer URL.

COM und .NET

Signatur: `ServerPath(string path)`

Java

Signatur: `public void setServerPath(string path)`

6.4.1.1.2.17 ServerPort

Definiert den Port des HTTP-Servers, über den der Dienst aufgerufen wird. Es muss sich um einen fixen und bekannten Port handeln, damit HTTP-Requests korrekt an den Dienst adressiert werden können. Der Input-Parameter ist eine Ganzzahl, die den Zugriffsport am HTTP-Server angibt. Verursacht eine

RaptorXMLException[333], wenn ein Fehler auftritt.

COM und .NET

Signatur: `ServerPort(int port)`

Java

Signatur: `public void setServerPort(int port)`

6.4.1.1.2.18 ServicePackVersion

Gibt die Service Pack-Version des Produkts als Ganzzahl zurück. *Beispiel:* Gibt für `RaptorXML Server 2025r2sp1(x64)` den Wert `1` (aus der Service Pack Versionsnummer `sp1`) zurück. Löst bei Fehler eine RaptorXMLException[333] aus.

COM und .NET

Signatur: `int ServicePackVersion()`

Java

Signatur: `public int getServicePackVersion()`

6.4.1.1.2.19 UserCatalog

Definiert den Pfad der benutzerdefinierten Katalogdatei in Form einer URL. Der bereitgestellte String muss eine absolute URL sein, die den genauen Pfad der zu verwendenden benutzerdefinierten Katalogdatei angibt.

COM und .NET

Signatur: `UserCatalog(string userCatalog)`

Java

Signatur: `public void setUserCatalog(string userCatalog)`

6.4.1.2 RaptorXMLException

Generiert eine Ausnahme, die Informationen über den bei der Verarbeitung aufgetretenen Fehler enthält. Der Parameter `message` enthält Informationen über den Fehler.

COM und .NET

Signatur: `RaptorXMLException(string message)`

Java

Signatur: `public void RaptorXMLException(string message)`

6.4.1.3 XMLDSig (für XML-Signaturen)

Mit Hilfe der Methoden der **`IXMLDSig/XMLDSig`**-Schnittstelle/Klasse können XML-Dokumente signiert, signierte Dokumente überprüft, zuvor signierte Dokumente, die geändert wurden, (mit einer neuen Signatur) aktualisiert und Signaturen entfernt werden.

Beachten Sie, dass der Name der Schnittstelle in der COM/.NET API anders lautet als der der Klasse in der Java API:

- In COM/.NET: **`IXMLDSig`**
- In Java: **`XMLDSig`**

6.4.1.3.1 Methoden

In diesem Abschnitt sind die Methoden der **`IXMLDSig`**-Schnittstelle (COM/.NET) und der **`XMLDSig`**-Klasse (Java) beschrieben.

6.4.1.3.1.1 ExecuteRemove

Entfernt die XML-Signatur der signierten XML-Datei und speichert das nicht signierte Dokument unter einem durch `outputPath`, einem String, der die URL des Dateipfads angibt, definierten Ausgabepfad. Das Ergebnis ist bei Erfolg `true`, bei Fehlschlag. `false`.

COM und .NET

Signatur: `boolean ExecuteRemove(string outputPath)`

Java

Signatur: `public boolean executeRemove(string outputPath)`

6.4.1.3.1.2 ExecuteSign

Signiert das XML-Dokument gemäß den definierten Signierungsoptionen (die in den Parametern `signatureType` und `canonicalizationMethod` angegeben werden; *verfügbare Werte siehe xmlsignature-sign CLI-Befehl*[208]). Die Ausgabedatei wird durch `outputPath`, einen String, der die URL der Ausgabedatei angibt, definiert. Das Ergebnis ist bei Erfolg `true`, bei Fehlschlag `false`.

COM und .NET

Signatur: `boolean ExecuteSign(string outputPath, string signatureType,string canonicalizationMethod)`

Java

Signatur: `public boolean executeSign(string outputPath, string signatureType,string canonicalizationMethod)`

6.4.1.3.1.3 ExecuteUpdate

Aktualisiert die XML-Signatur in der signierten XML-Datei. Wenn das Dokument geändert wurde, ist die aktualisierte XML-Signatur eine andere. Andernfalls ist die aktualisierte Signatur mit der vorherigen Signatur identisch. Die Ausgabedatei wird durch `outputPath`, einen String, der die URL der Datei mit der aktualisierten Signatur angibt, definiert. Das Ergebnis ist bei Erfolg `true`, bei Fehlschlag `false`.

Es müssen entweder die Eigenschaft (i) HMAC secret key[338] oder (ii) die Eigenschaften certificate-name[336] und certificate-store[336] definiert werden. Wenn die certifiate-Optionen definiert sind, müssen diese mit denjenigen übereinstimmen, die zuvor zum Signieren des XML-Dokuments verwendet wurden. (Beachten Sie, dass die Option certificate-store derzeit unter Linux und macOS nicht unterstützt wird.)

COM und .NET

Signatur: `boolean ExecuteUpdate(string outputPath)`

Java

Signatur: `public boolean executeUpdate(string outputPath)`

6.4.1.3.1.4 ExecuteVerify

Gibt das Ergebnis der Signaturüberprüfung zurück: **true** bei Erfolg, andernfals **false**.

COM und .NET

Signatur: `boolean` **`ExecuteVerify()`**

Java

Signatur: **`public`** `boolean` **`executeVerify()`**

6.4.1.3.2 Eigenschaften

In diesem Abschnitt sind die Eigenschaften der **IXMLDSig**-Schnittstelle (COM/.NET) und der **XMLDSig**-Klasse (Java) beschrieben.

6.4.1.3.2.1 AbsoluteReferenceUri

Definiert, ob die URI des signierten Dokuments als absoluter (`true`) oder relativer Pfad (`false`) gelesen werden soll. Die Standardeinstellung ist `false`.

COM und .NET

Signatur: **`AbsoluteReferenceUri(boolean absoluteuri)`**

Java

Signatur: **`public`** `void` **`setAbsoluteReferenceUri(boolean absoluteuri)`**

6.4.1.3.2.2 AppendKeyInfo

Definiert, ob das Element `KeyInfo` in die Signatur inkludiert werden soll oder nicht. Der Standardwert ist `false`.

COM und .NET

Signatur: **`AppendKeyInfo(boolean include)`**

Java

Signatur: `public void setAppendKeyInfo(boolean inlude)`

6.4.1.3.2.3 CertificateName

Der Name des zum Signieren verwendeten Zertifikats.

Windows
Dies ist der **Subject-**Name eines Zertifikats aus dem ausgewählten `--certificate-store` (Zertifikatspeicher).

Beispiel zum Aufliste der Zertifikate (unter PowerShell)

```
% ls cert://CurrentUser/My
PSParentPath: Microsoft.PowerShell.Security\Certificate::CurrentUser\My
Thumbprint Subject
---------- -------
C9DF64BB0AAF5FA73474D78B7CCFFC37C95BFC6C CN=certificate1
... CN=...
```

Beispiel: `--certificate-name==certificate1`

Linux/MacOS
`--certname` definiert den Dateinamen eines PEM-kodierten X.509v3-Zertifikats mit dem Private Key. Solche Dateien haben die Erweiterung `.pem`.

Beispiel: `--certificate-name==/path/to/certificate1.pem`

COM und .NET

Signatur: `CertificateName(string name)`

Java

Signatur: `public void setCertificateName(string name)`

6.4.1.3.2.4 CertificateStore

Der Pfad, unter dem das mit `--certificate-name` definierte Zertifikat gespeichert ist.

Windows
Der Name eines Zertifikatspeichers unter `cert://CurrentUser`. Die verfügbaren Zertifikatspeicher können mit

Hilfe von `% ls cert://CurrentUser/` (unter PowerShell) aufgelistet werden. Die Zertifikate würden anschließend folgendermaßen aufgelistet:

```
Name : TrustedPublisher
Name : ClientAuthIssuer
Name : Root
Name : UserDS
Name : CA
Name : ACRS
Name : REQUEST
Name : AuthRoot
Name : MSIEHistoryJournal
Name : TrustedPeople
Name : MyCertStore
Name : Local NonRemovable Certificates
Name : SmartCardRoot
Name : Trust
Name : Disallowed
```

Beispiel: `--certificate-store==MyCertStore`

Linux/MacOS
Die Option `--certstore` wird derzeit nicht unterstützt.

COM und .NET

Signatur: `CertificateStore(string filelocation)`

Java

Signatur: `public void setCertificateStore(string filelocation)`

6.4.1.3.2.5 DigestMethod

Der zur Berechnung des Digest-Werts an der XML-Input-Datei angewendete Algorithmus. Verfügbare Werte sind: `sha1|sha256|sha384|sha512`.

COM und .NET

Signatur: `DigestMethod(string algo)`

Java

Signatur: `public void setDigestMethod(string algo)`

6.4.1.3.2.6 HMACOutputLength

Kürzt die Ausgabe des HMAC-Algorithmus auf `length` Bits. Falls definiert, muss dieser Wert

- ein Vielfaches von 8 sein
- größer als 80 sein
- mehr als die Hälfte der Ausgabelänge des zugrunde liegenden Hash-Algorithmus betragen

COM und .NET

Signatur: **`HMACOutputLength(int length)`**

Java

Signatur: **`public void setHMACOutputLength(int length)`**

6.4.1.3.2.7 HMACSecretKey

Der Shared Secret HMAC-Schlüssel; muss mindestens sechs Zeichen lang sein.

COM und .NET

Signatur: **`HMACSecretKey(string key)`**

Java

Signatur: **`public void setHMACSecretKey(string key)`**

6.4.1.3.2.8 InputXMLFileName

Definiert den Pfad des zu verarbeitenden XML-Dokuments in Form einer URL. Der bereitgestellte String muss eine absolute URL sein, die den genauen Pfad der XML-Datei angibt.

COM und .NET

Signatur: **`InputXMLFileName(string filepath)`**

Java

Signatur: **`public void setInputXMLFileName(string filepath)`**

6.4.1.3.2.9 LastErrorMessage

Ruft einen String auf, der die letzte Fehlermeldung aus dem RaptorXML-Prozessor darstellt, auf.

COM und .NET

Signatur: `string` **`LastErrorMessage()`**

Java

Signatur: **`public`** `string` **`getLastErrorMessage()`**

6.4.1.3.2.10 SignatureMethod

Definiert, welcher Algorithmus zum Generieren der Signatur verwendet werden soll.

Wenn ein Zertifikat verwendet wird
Wenn ein Zertifikat definiert ist, dann ist `SignatureMethod` optional und der Wert dieses Parameters wird vom Zertifikat abgeleitet. Wenn die Option definiert ist, muss sie mit dem vom Zertifikat verwendeten Algorithmus übereinstimmen.
Beispiel: `rsa-sha256`.

Wenn --hmac-secret-key verwendet wird
Wenn `HMACSecretKey` verwendet wird, ist diese `SignatureMethod` obligatorisch. Der Wert muss einer der unterstützten HMAC-Algorithmen sein:

- `hmac-sha256`
- `hmac-sha386`
- `hmac-sha512`
- hmac-sha1 (soll laut Spezifikation nicht verwendet werden)

Beispiel: **`hmac-sha256`**

COM und .NET

Signatur: **`SignatureMethod(string algo)`**

Java

Signatur: **`public`** `void` **`setSignatureMethod(string algo)`**

6.4.1.3.2.11 Transforms

Definiert, welche XML-Signaturtransformation auf das Dokument angewendet werden soll. Die unterstützten Werte sind die folgenden:

- `REC-xml-c14n-20010315` für Canonical XML 1.0 (Kommentare weglassen)
- `xml-c14n11` für Canonical XML 1.1 (Kommentare weglassen)
- `xml-exc-c14n#` für Exclusive XML Canonicalization 1.0 (Kommentare weglassen)
- `REC-xml-c14n-20010315#WithComments` für Canonical XML 1.0 (mit Kommentaren)
- `xml-c14n11#WithComments` for Canonical XML 1.1 (mit Kommentaren)
- `xml-exc-c14n#WithComments` for Exclusive XML Canonicalization 1.0 (mit Kommentaren)
- `base64`
- `strip-whitespaces` Altova-Erweiterung

COM und .NET

Signatur: `Transforms(string value)`

Java

Signatur: `public void setTransforms(string value)`

6.4.1.3.2.12 WriteDefaultAttributes

Gibt an, ob Standardattributwerte aus der DTD im signierten Dokument inkludiert werden sollen.

COM und .NET

Signatur: `WriteDefaultAttributes(boolean write)`

Java

Signatur: `public void setWriteDefaultAttributes(boolean write)`

6.4.1.4 XMLValidator

Die **`IXMLValidator/XMLValidator`**-Schnittstelle/Klasse bietet die folgenden Methoden, um (i) die verschiedenen Arten von Dokumenten zu validieren, (ii) Dokumente auf ihre Wohlgeformtheit zu prüfen und (iii) ein Avro-Schema aus einer Avro-Binärdatei zu extrahieren. Über ein Python-Skript können Sie zusätzliche Verarbeitungen bereitstellen.

Beachten Sie, dass die Schnittstelle in der COM/.NET API einen anderen Namen hat als die Klasse in der Java API:

- In COM/.NET: **`IXMLValidator`**
- In Java: **`XMLValidator`**

6.4.1.4.1 Methoden

In diesem Abschnitt sind die Methoden der **`IXMLValidator`**-Schnittstelle (COM/.NET) und der **`XMLValidator`**-Klasse (Java) beschrieben.

6.4.1.4.1.1 AddPythonScriptFile

Definiert den Pfad der Python Skriptdatei, die eine zusätzliche Verarbeitung der zur Validierung gesendeten Datei bereitstellt. Der bereitgestellte String muss eine absolute URL des Python-Skripts sein. Das Python-Skript wird mit einem mit RaptorXML Server gebündelten Python-Paket verarbeitet. Das gebündelte Paket ist die Version 3.11.8.

COM und .NET

Signatur: **`AddPythonScriptFile(`**`string` **`filepath)`**

Java

Signatur: **`public`** `void` **`addPythonScriptFile(`**`string` **`filepath)`**

6.4.1.4.1.2 ClearPythonScriptFile

Löscht mit der Methode **`AddPythonScriptFile`** oder der Eigenschaft **`PythonScriptFile`** hinzugefügte Python-Skript-Dateien.

COM und .NET

Signatur: **`ClearPythonScriptFile()`**

Java

Signatur: **`public`** `void` **`clearPythonScriptFile()`**

6.4.1.4.1.3 ExtractAvroSchema

Extrahiert ein Avro-Schema aus einer Binärdatei. Der Parameter `outputPath` ist eine absolute URL, die den Ausgabeordner definiert. Das Ergebnis ist `true` bei Erfolg, `false` bei einem Fehlschlag. Bei Auftreten eines Fehlers wird eine **RaptorXMLException** 333 ausgelöst. Zusätzliche Informationen können Sie mit der Methode `LastErrorMessage` aufrufen.

COM und .NET

Signatur: **ExtractAvroSchema(**string **outputPath)**

Java

Signatur: **public** void **extractAvroSchema(**string **outputPath)**

6.4.1.4.1.4 IsValid

Gibt das Ergebnis der Validierung des XML-Dokuments, Schema-Dokuments oder DTD-Dokuments zurück. Welcher Dokumenttyp validiert wird, wird durch den Parameter `type` definiert, der ein ENUMValidationType 380 Literal als Wert erhält. Das Ergebnis ist bei Erfolg `true`, bei Fehlschlag `false`. Wenn ein Fehler auftritt, wird eine **RaptorXMLException** 333 ausgegeben. Zusätzliche Informationen können Sie mit der Methode `LastErrorMessage` aufrufen.

COM und .NET

Signatur: boolean **IsValid(**ENUMValidationType 380 **type)**

Java

Signatur: **public** boolean **isValid(**ENUMValidationType 380 **type)**

6.4.1.4.1.5 IsWellFormed

Gibt das Ergebnis der Wohlgeformtheitsprüfung des XML- oder DTD-Dokuments zurück. Welcher Dokumenttyp überprüft wird, wird durch den Parameter `type` angegeben. Dieser Parameter erhält ein ENUMWellformedCheckType 382 als Wert. Das Ergebnis ist bei Erfolg `true`, bei Fehlschlag `false`. Bei einem Fehler wird eine **RaptorXMLException** 333 ausgegeben.Zusätzliche Informationen können Sie mit der Methode `LastErrorMessage` aufrufen.

COM und .NET

Signatur: boolean **isWellFormed(**ENUMWellformedCheckType 380 **type)**

Java

Signatur: `public boolean isWellFormed(ENUMWellformedCheckType 380 type)`

6.4.1.4.2 Eigenschaften

In diesem Abschnitt sind die Eigenschaften der **IXMLValidator**-Schnittstelle (COM/.NET) und der **XMLValidator**-Klasse (Java) beschrieben.

6.4.1.4.2.1 AssessmentMode

Definiert den XML-Validierungsmodus (`Strict/Lax`), der durch ein ENUMAssessmentMode 376-Literal definiert wird.

COM und .NET

Signatur: `AssessmentMode(ENUMAssessmentMode 376 mode)`

Java

Signatur: `public void setAssessmentMode(ENUMAssessmentMode 376 mode)`

6.4.1.4.2.2 AvroSchemaFileName

Definiert den Pfad des externen Avro-Schemas, das verwendet werden soll. Der bereitgestellte String muss eine absolute URL sein, die den exakten Pfad der Avro-Schema-Datei angibt.

COM und .NET

Signatur: `AvroSchemaFileName(string url)`

Java

Signatur: `public void setAvroSchemaFileName(string url)`

6.4.1.4.2.3 AvroSchemaFromText

Stellt einen String bereit, der der Inhalt des Avro-Schema-Dokuments, das verwendet werden soll, ist.

COM und .NET

Signatur: `AvroSchemaFromText(string avroschema)`

Java

Signatur: `public void setAvroSchemaFromText(string avroschema)`

6.4.1.4.2.4 DTDFileName

Definiert den Pfad des für die Validierung zu verwendenden DTD-Dokuments in Form einer URL. Der angegebene String muss eine absolute URL sein, die den genauen Pfad der zu verwendenden DTD definiert.

COM und .NET

Signatur: `DTDFileName(string url)`

Java

Signatur: `public void setDTDFileName(string url)`

6.4.1.4.2.5 DTDFromText

Liefert einen String, der der Textinhalt des für die Validierung zu verwendenden DTD-Dokuments ist.

COM und .NET

Signatur: `DTDFromText(string dtdtext)`

Java

Signatur: `public void setDTDFromText(string dtdtext)`

6.4.1.4.2.6 EnableNamespaces

Aktiviert die Namespace-fähige Verarbeitung. Dies ist nützlich, um die XML-Instanz auf Fehler infolge falscher Namespaces zu überprüfen. Der Wert `true` aktiviert die Namespace-fähige Verarbeitung; `false` deaktiviert sie. Der Standardwert ist `false`.

COM und .NET

Signatur: **`EnableNamespaces(`**`boolean` **`enableNS)`**

Java

Signatur: **`public`** `void` **`setEnableNamespaces(`**`boolean` **`enableNS)`**

6.4.1.4.2.7 InputFileArray

Liefert ein Array der URLs der XML-Dateien, die als Input-Daten verwendet werden sollen. Der Array liefert ein Objekt, das die Strings der absoluten URLs der einzelnen Input-Dateien enthält.

COM und .NET

Signatur: **`InputFileArray(`**`object` **`fileArray)`**

Java

Signatur: **`public`** `void` **`setInputFileArray(`**`object` **`fileArray)`**

6.4.1.4.2.8 InputFileName

Definiert den Pfad der zu verarbeitenden Input-Datendatei als URL. Der bereitgestellte String muss eine absolute URL sein, die den Pfad der zu verwendenden Datei angibt.

COM und .NET

Signatur: **`InputFileName(`**`string` **`filepath)`**

Java

Signatur: **`public`** `void` **`setInputFileName(`**`string` **`filepath)`**

6.4.1.4.2.9 InputFromText

Stellt einen String bereit, der der Textinhalt des zu verarbeitenden Dokuments ist.

COM und .NET

Signatur: `InputFromText(string doc)`

Java

Signatur: `public void setInputFromText(string doc)`

6.4.1.4.2.10 InputTextArray

Liefert ein Array der URLs der Textdateien, die als Input-Daten verwendet werden sollen. Die Eigenschaft liefert ein Objekt, das die absoluten URLs der einzelnen Textdateien in Form von Strings enthält.

COM und .NET

Signatur: `InputTextArray(object textfileArray)`

Java

Signatur: `public void setInputTextArray(object textfileArray)`

6.4.1.4.2.11 InputXMLFileName

Definiert den Pfad des zu verarbeitenden XML-Dokuments in Form einer URL. Der bereitgestellte String muss eine absolute URL sein, die den genauen Pfad der XML-Datei angibt.

COM und .NET

Signatur: `InputXMLFileName(string url)`

Java

Signatur: `public void setInputXMLFileName(string url)`

6.4.1.4.2.12 InputXMLFromText

Liefert den Inhalt des zu verarbeitenden XML-Dokuments. Der bereitgestellte String ist der Inhalt des zu verarbeitenden XML-Dokuments.

COM und .NET

Signatur: `InputXMLFromText(string xml)`

Java

Signatur: `public void setInputXMLFromText(string xml)`

6.4.1.4.2.13 Json5

Bei `true` wird die JSON 5-Unterstützung aktiviert.

COM und .NET

Signatur: `Json5(boolean json5)`

Java

Signatur: `public void setJson5(boolean json5)`

6.4.1.4.2.14 JSONSchemaFileName

Definiert den Pfad der JSON-Schema-Datei, anhand der das JSON-Instanzdokument validiert werden soll, in Form einer URL. Der bereitgestellte String muss eine absolute URL sein, die den genauen Pfad zur JSON-Schema-Datei angibt.

COM und .NET

Signatur: `JSONSchemaFileName(string url)`

Java

Signatur: `public void setJSONSchemaFileName(string url)`

6.4.1.4.2.15 JSONSchemaFromText

Liefert einen String, der den Textinhalt des JSON-Schema-Dokuments, anhand dessen das JSON-Instanzdokument validiert wird, darstellt.

COM und .NET

Signatur: `JSONSchemaFromText(string jsonschema)`

Java

Signatur: `public void setJSONSchemaFromText(string jsonschema)`

6.4.1.4.2.16 LastErrorMessage

Ruft einen String auf, der die letzte Fehlermeldung aus dem RaptorXML-Prozessor darstellt, auf.

COM und .NET

Signatur: `string LastErrorMessage()`

Java

Signatur: `public string getLastErrorMessage()`

6.4.1.4.2.17 ParallelAssessment

Aktiviert/deaktiviert die parallele Validierung von Schemas[252].

COM und .NET

Signatur: `ParallelAssessment(boolean enable)`

Java

Signatur: `public void setParallelAssessment(boolean enable)`

6.4.1.4.2.18 PythonScriptFile

Definiert den Pfad der Python Skriptdatei, die eine zusätzliche Verarbeitung der zur Validierung gesendeten Datei bereitstellt. Der bereitgestellte String muss eine absolute URL des Python-Skripts sein. Das Python-Skript wird mit einem mit RaptorXML Server gebündelten Python-Paket verarbeitet. Das gebündelte Paket ist die Version 3.11.8.

COM und .NET

Signatur: `PythonScriptFile(string filepath)`

Java

Signatur: `public void setPythonScriptFile(string filepath)`

6.4.1.4.2.19 SchemaFileArray

Liefert die Sammlung der XML-Dateien, die als externe XML-Schemas verwendet werden. Die Dateien werden anhand ihrer URLs identifiziert. Der Input ist eine Sammlung von Strings, von denen jeder die absolute URL einer XML-Schema-Datei ist.

COM und .NET

Signatur: `SchemaFileArray(object urlArray)`

Java

Signatur: `public void setSchemaFileArray(object urlArray)`

6.4.1.4.2.20 SchemaFileName

Definiert den Pfad des XML-Schema-Dokuments, anhand dessen die Validierung durchgeführt werden soll, in Form einer URL. Der gelieferte String muss eine absolute URL sein, die den genauen Pfad der XML-Schema-Datei angibt.

COM und .NET

Signatur: `SchemaFileName(string filepath)`

Java

Signatur: `public void setSchemaFileName(string filepath)`

6.4.1.4.2.21 SchemaFromText

Liefert den Inhalt des zu verwendenden XML-Schema-Dokuments. Der gelieferte String ist der Inhalt des zu verwendenden XML-Schema-Dokuments.

COM und .NET

Signatur: `SchemaFileName(string xsdText)`

Java

Signatur: `public void setSchemaFileName(string xsdText)`

6.4.1.4.2.22 SchemaImports

Definiert, wie Schemaimporte auf Basis der Attributwerte der `xs:import` Elemente zu behandeln sind. Die Art der Behandlung wird durch das bereitgestellte `ENUMSchemaImports`-Literal definiert.

COM und .NET

Signatur: `SchemaImports(ENUMSchemaImports 378 importOption)`

Java

Signatur: `public void setSchemaImports(ENUMSchemaImports 378 importOption)`

6.4.1.4.2.23 SchemalocationHints

Definiert, welcher Mechanismus zum Auffinden des Schemas verwendet werden soll. Der Mechanismus wird durch das ausgewählte `ENUMLoadSchemalocation` Literal definiert.

COM und .NET

Signatur: `SchemalocationHints(ENUMLoadSchemalocation 377 hint)`

Java

Signatur: `public void setSchemalocationHints(ENUMLoadSchemalocation 377 hint)`

6.4.1.4.2.24 SchemaMapping

Definiert, welches Mapping zum Auffinden des Schemas verwendet werden soll. Das Mapping wird durch das ausgewählte `ENUMSchemaMapping`-Literal definiert.

COM und .NET

Signatur: **`SchemaMapping(`**ENUMSchemaMapping[379] **`mappingOption)`**

Java

Signatur: **`public`** `void` **`setSchemaMapping(`**ENUMSchemaMapping[379] **`mappingOption)`**

6.4.1.4.2.25 SchemaTextArray

Liefert den Inhalt mehrerer XML-Schema-Dokumente. Der Input ist eine Sammlung von Strings, von denen jede der Inhalt eines XML-Schema-Dokuments ist.

COM und .NET

Signatur: **`SchemaTextArray(`**`object` **`schemaDocs)`**

Java

Signatur: **`public`** `void` **`setSchemaTextArray(`**`object` **`schemaDocs)`**

6.4.1.4.2.26 Streaming

Aktiviert die Streaming-Validierung. Im Streaming-Modus werden möglichst wenige Daten im Arbeitsspeicher behalten, wodurch die Verarbeitung beschleunigt wird. Ein Wert `true` aktiviert das Streaming; `false` deaktiviert es. Der Standardwert ist `true`.

COM und .NET

Signatur: **`Streaming(`**`boolean` **`enable)`**

Java

Signatur: **`public`** `void` **`setStreaming(`**`boolean` **`enable)`**

6.4.1.4.2.27 XincludeSupport

Aktiviert oder deaktiviert die Verwendung von `XInclude` Elementen. Der Wert `true` aktiviert die XInclude-Unterstützung; `false` deaktiviert sie. Der Standardwert ist `false`.

COM und .NET

Signatur: **`XincludeSupport(boolean xinclude)`**

Java

Signatur: **`public void setXincludeSupport(boolean xinclude)`**

6.4.1.4.2.28 XMLValidationMode

Definiert den XML-Validierungsmodus, welcher ein Enumerationsliteral von ENUMXMLValidationMode[382] ist, welches festlegt, ob die Gültigkeit oder Wohlgeformtheit geprüft wird.

COM und .NET

Signatur: **`XMLValidationMode(ENUMXMLValidationMode[382] valMode)`**

Java

Signatur: **`public void setXMLValidationMode(ENUMXMLValidationMode[382] valMode)`**

6.4.1.4.2.29 XSDVersion

Definiert die XML-Schema-Version, anhand welcher das XML-Dokument validiert wird. Der Wert ist ein Enumerationsliteral von ENUMXSDVersion[385].

COM und .NET

Signatur: **`XSDVersion(ENUMXSDVersion[385] version)`**

Java

Signatur: **`public void setXSDVersion(ENUMXSDVersion[385] version)`**

6.4.1.5 XQuery

Die **IXQuery/XQuery**-Schnittstelle/Klasse bietet Methoden, um (i) XQuery-Dokumente und -Updates auszuführen und (ii) im Zusammenhang mit XQuery stehende Dokumente zu validieren. Sie können auch mittels externer Variablen Daten für die Ausführung angeben.

Beachten Sie, dass der Name der Schnittstelle in der COM/.NET API nicht derselbe ist wie der der Klasse in der Java API:

- In COM/.NET: **IXQuery**
- In Java: **XQuery**

6.4.1.5.1 Methoden

In diesem Abschnitt sind die Methoden der **IXQuery**-Schnittstelle (COM/.NET) und der **XQuery**-Klasse (Java) beschrieben.

6.4.1.5.1.1 AddExternalVariable

Fügt den Namen und Wert einer neuen externen Variablen hinzu. Jede externe Variable und ihr Wert müssen in einem separaten Methodenaufruf definiert werden. Variablen müssen im XQuery-Dokument (mit einer optionalen Typdeklaration) deklariert werden. Setzen Sie den Variablenwert in einfache Anführungszeichen, wenn der Variablenwert ein String ist. Der Parameter `name` enthält den Namen der Variablen, welcher ein QName in Form eines String ist. Der Parameter `value` enthält den Wert der Variablen als String.

COM und .NET

Signatur: **AddExternalVariable(**string **name,** string **value)**

Java

Signatur: **public** void **addExternalVariable(**string **name,** string **value)**

6.4.1.5.1.2 ClearExternalVariableList

Löscht die Liste der mit der AddExternalVariable 353 Methode erstellten externen Variablen.

COM und .NET

Signatur: **ClearExternalVariableList()**

Java

Signatur: `public void clearExternalVariableList()`

6.4.1.5.1.3 Execute

Führt die XQuery-Transformation anhand der in `EngineVersion`[357] Eigenschaft genannten XQuery-Version aus und speichert das Ergebnis in der im Parameter `outputFile` genannten Ausgabedatei. Der Parameter ist ein String, der den Pfad (Pfad und Dateinamen) der Ausgabedatei liefert. Bei erfolgreicher Ausführung wird der Boolesche Wert `true` zurückgegeben, bei Fehlschlag der Wert `false`. Bei Auftreten eines Fehlers wird eine `RaptorXMLException`[333] ausgelöst. Mit Hilfe der `LastErrorMessage` Methode können Sie zusätzliche Informationen aufrufen.

COM und .NET

Signatur: `boolean Execute(string outputFile)`

Java

Signatur: `public boolean Execute(string outputFile)`

6.4.1.5.1.4 ExecuteAndGetResultAsString

Führt das XQuery Updaten anhand der in der Eigenschaft `EngineVersion`[357] genannten XQuery Update-Spezifikation aus und gibt das Ergebnis als String zurück. Mit dieser Methode werden keine zusätzlichen Ergebnisdateien wie z.B. Diagramme oder sekundäre Ergebnisse erzeugt. Sie enthält auch keine Binärdateiergebnisse wie z.B. `.docx` OOXML-Dateien. Falls zusätzliche Ausgabedateien benötigt werden, verwenden Sie die `Execute`[354]-Methode.

COM und .NET

Signatur: `string ExecuteAndGetResultAsString()`

Java

Signatur: `public string executeAndGetResultAsString()`

6.4.1.5.1.5 ExecuteUpdate

Führt das XQuery Update entsprechend der in der Eigenschaft `XQueryUpdateVersion`[364] genannten XQuery Update-Spezifikation aus und speichert das Ergebnis in der im Parameter `outputFile` angegebenen Ausgabedatei. Der Parameter ist ein String, der den Pfad (Pfad und Dateinamen) der Ausgabedatei angibt. Bei erfolgreicher Ausführung wird der Boolesche Wert `true` zurückgegeben, bei Fehlschlag der Wert `false`. Bei Auftreten eines Fehlers wird eine `RaptorXMLException`[333] ausgelöst. Mit Hilfe der `LastErrorMessage` Methode können Sie zusätzliche Informationen aufrufen.

COM und .NET

Signatur: `boolean` **`ExecuteUpdate(`**`string` **`outputFile)`**

Java

Signatur: **`public`** `boolean` **`executeUpdate(`**`string` **`outputFile)`**

6.4.1.5.1.6 ExecuteUpdateAndGetResultAsString

Führt das XQuery Update anhand der in der Eigenschaft `XQueryUpdateVersion`[364] genannten XQuery Update-Spezifikation aus und gibt das Ergebnis als String zurück. Mit dieser Methode werden keine zusätzlichen Ergebnisdateien wie z.B. Diagramme oder sekundäre Ergebnisse erzeugt. Sie enthält auch keine Binärdateiergebnisse wie z.B. `.docx` OOXML-Dateien.

COM und .NET

Signatur: `string` **`ExecuteUpdateAndGetResultAsString()`**

Java

Signatur: **`public`** `string` **`executeUpdateAndGetResultAsString()`**

6.4.1.5.1.7 IsValid

Gibt das Ergebnis der Validierung des XQuery-Dokuments, die anhand der in Eigenschaft `EngineVersion`[357] genannten XQuery-Spezifikation durchgeführt wurde, zurück. Das Ergebnis ist bei Erfolg `true`, bei Fehlschlag `false`. Bei Auftreten eines Fehlers wird eine `RaptorXMLException`[333] ausgelöst. Mit Hilfe der Eigenschaft `LastErrorMessage` können Sie zusätzliche Informationen aufrufen.

COM und .NET

Signatur: `boolean` **`IsValid()`**

Java

Signatur: `public boolean isValid()`

6.4.1.5.1.8 IsValidUpdate

Gibt das Ergebnis der Validierung des XQuery Update-Dokuments, die anhand der in Eigenschaft `XQueryUpdateVersion`[364] genannten XQuery Update-Spezifikation durchgeführt wurde, zurück. Das Ergebnis ist bei Erfolg `true`, bei Fehlschlag `false`. Bei Auftreten eines Fehlers wird eine `RaptorXMLException`[333] ausgelöst. Mit Hilfe der Eigenschaft `LastErrorMessage` können Sie zusätzliche Informationen aufrufen.

COM und .NET

Signatur: `boolean IsValidUpdate()`

Java

Signatur: `public boolean isValidUpdate()`

6.4.1.5.2 Eigenschaften

In diesem Abschnitt sind die Eigenschaften der **`IXQuery`**-Schnittstelle (COM/.NET) und der **`XQuery`**-Klasse (Java) beschrieben.

6.4.1.5.2.1 AdditionalOutputs

Gibt die zusätzlichen Ausgabedokumente des zuletzt ausgeführten Auftrags zurück.

COM und .NET

Signatur: `string AdditionalOutputs()`

Java

Signatur: `public string getAdditionalOutputs()`

6.4.1.5.2.2 ChartExtensionsEnabled

Aktiviert bzw. deaktiviert Altova-Diagrammerweiterungsfunktionen. Ein Wert `true` aktiviert Diagrammerweiterungen; `false` deaktiviert sie. Der Standardwert ist `true`.

COM und .NET

Signatur: **`ChartExtensionsEnabled(boolean enable)`**

Java

Signatur: **`public void setChartExtensionsEnabled(boolean enable)`**

6.4.1.5.2.3 DotNetExtensionsEnabled

Aktiviert oder deaktiviert .NET-Erweiterungsfunktionen. Ein Wert `true` aktiviert .NET-Erweiterungen; `false` deaktiviert sie. Der Standardwert ist `true`.

COM und .NET

Signatur: **`DotNetExtensionsEnabled(boolean enable)`**

Java

Signatur: **`public void setDotNetExtensionsEnabled(boolean enable)`**

6.4.1.5.2.4 EngineVersion

Definiert, welche XQuery-Version verwendet werden soll. Die Eigenschaft ist ein ENUMXQueryVersion[384] Literal.

COM und .NET

Signatur: **`EngineVersion(`**ENUMXQueryVersion[384] **`version)`**

Java

Signatur: **`public void setEngineVersion(`**ENUMXQueryVersion[384] **`version)`**

6.4.1.5.2.5 IndentCharacters

Definiert den in der Ausgabe als Einrückung zu verwendenden Zeichenstring.

COM und .NET

Signatur: `IndentCharacters(string indentChars)`

Java

Signatur: `public void setIndentCharacters(string indentChars)`

6.4.1.5.2.6 InputXMLFileName

Definiert den Pfad des zu verarbeitenden XML-Dokuments in Form einer URL. Der bereitgestellte String muss eine absolute URL sein, die den genauen Pfad der XML-Datei angibt.

COM und .NET

Signatur: `InputXMLFileName(string url)`

Java

Signatur: `public void setInputXMLFileName(string url)`

6.4.1.5.2.7 InputXMLFromText

Liefert den Inhalt des zu verarbeitenden XML-Dokuments. Der bereitgestellte String ist der Inhalt des zu verarbeitenden XML-Dokuments.

COM und .NET

Signatur: `InputXMLFromText(string xml)`

Java

Signatur: `public void setInputXMLFromText(string xml)`

6.4.1.5.2.8 JavaBarcodeExtensionLocation

Definiert den Pfad der Java-Barcode-Erweiterungsdatei. Nähere Informationen dazu finden Sie im Abschnitt Barcode-Erweiterungsfunktionen von Altova[526]. Der bereitgestellte String muss eine absolute URL sein, die den Basispfad der zu verwendenden Datei angibt.

COM und .NET

Signatur: `JavaBarcodeExtensionLocation(string url)`

Java

Signatur: `public void setJavaBarcodeExtensionLocation(string url)`

6.4.1.5.2.9 JavaExtensionsEnabled

Aktiviert oder deaktiviert Java-Erweiterungsfunktionen. Der Wert `true` aktiviert Java-Erweiterungen; `false` deaktiviert sie. Der Standardwert ist `true`.

COM und .NET

Signatur: `JavaExtensionsEnabled(boolean enable)`

Java

Signatur: `public void setJavaExtensionsEnabled(boolean enable)`

6.4.1.5.2.10 KeepFormatting

Definiert, ob die Formatierung des Originaldokuments (so weit wie möglich) beibehalten werden soll oder nicht. Beim Wert `true` wird die Formatierung beibehalten, bei `false` wird sie nicht beibehalten. Der Standardwert ist `true`.

COM und .NET

Signatur: `KeepFormatting(boolean keep)`

Java

Signatur: `public void setKeepFormatting(boolean keep)`

6.4.1.5.2.11 LastErrorMessage

Ruft einen String auf, der die letzte Fehlermeldung aus dem RaptorXML-Prozessor darstellt, auf.

COM und .NET

Signatur: `string LastErrorMessage()`

Java

Signatur: `public string getLastErrorMessage()`

6.4.1.5.2.12 LoadXMLWithPSVI

Aktiviert die Validierung von XML-Dateien und generiert eine Post-Schema-Validierungsinfo für diese. Mit dem Wert `true` wird die XML-Validierung aktiviert und eine Post-Schema-Validierungsinfo für die XML-Dateien generiert; `false` deaktiviert die Validierung. Standardwert ist `true`.

COM und .NET

Signatur: `LoadXMLWithPSVI(boolean enable)`

Java

Signatur: `public void setLoadXMLWithPSVI(boolean enable)`

6.4.1.5.2.13 MainOutput

Gibt die Hauptausgabe des zuletzt ausgeführten Auftrags zurück.

COM und .NET

Signatur: `string MainOutput()`

Java

Signatur: `public string getMainOutput()`

6.4.1.5.2.14 OutputEncoding

Definiert die Kodierung für das Ergebnisdokument. Verwenden Sie den Namen einer offiziellen IANA-Kodierung wie z.B. `UTF-8`, `UTF-16`, `US-ASCII`, `ISO-8859-1` als String.

COM und .NET

Signatur: `OutputEncoding(string encoding)`

Java

Signatur: `public void setOutputEncoding(string encoding)`

6.4.1.5.2.15 OutputIndent

Aktiviert bzw. deaktiviert die Einrückung des Ausgabedokuments. Der Wert `true` aktiviert die Einrückung; `false` deaktiviert sie.

COM und .NET

Signatur: `OutputIndent(boolean outputIndent)`

Java

Signatur: `public void setOutputIndent(boolean outputIndent)`

6.4.1.5.2.16 OutputMethod

Definiert die Serialisierung des Ausgabedokuments. Gültige Werte sind: `xml|xhtml|html|text`. Der Standardwert ist `xml`.

COM und .NET

Signatur: `OutputMethod(string format)`

Java

Signatur: `public void setOutputMethod(string format)`

6.4.1.5.2.17 OutputOmitXMLDeclaraton

Aktiviert/deaktiviert den Einschluss der XML-Deklaration in das Ergebnisdokument. Der Wert `true` lässt die Deklaration weg; `false` inkludiert sie. Der Standardwert ist `false`.

COM und .NET

Signatur: **`OutputOmitXMLDeclaration(`**`boolean` **`omitDeclaration)`**

Java

Signatur: **`public`** `void` **`setOutputOmitXMLDeclaration(`**`boolean` **`omitDeclaration)`**

6.4.1.5.2.18 UpdatedXMLWriteMode

Definiert, wie Aktualisierungen der XML-Datei gehandhabt werden. Die Eigenschaftswert ist ein ENUMXQueryUpdatedXML[383] Literal.

COM und .NET

Signatur: **`UpdateXMLWriteMode(`**ENUMXQueryUpdatedXML[383] **`updateMode)`**

Java

Signatur: **`public`** `void` **`setUpdateXMLWriteMode(`**ENUMXQueryUpdatedXML[383] **`updateMode)`**

6.4.1.5.2.19 XincludeSupport

Aktiviert oder deaktiviert die Verwendung von `XInclude` Elementen. Der Wert `true` aktiviert die XInclude-Unterstützung; `false` deaktiviert sie. Der Standardwert ist `false`.

COM und .NET

Signatur: **`XincludeSupport(`**`boolean` **`xinclude)`**

Java

Signatur: **`public`** `void` **`setXincludeSupport(`**`boolean` **`xinclude)`**

6.4.1.5.2.20 XMLValidationErrorsAsWarnings

Aktiviert/Deaktiviert die Behandlung von XML-Validierungsfehlern als Warnungen. Erhält den Booleschen Wert `true` oder `false`.

COM und .NET

Signatur: **`XMLValidationErrorsAsWarnings(boolean enable)`**

Java

Signatur: **`public void setXMLValidationErrorsAsWarnings(boolean enable)`**

6.4.1.5.2.21 XMLValidationMode

Definiert den XML-Validierungsmodus, welcher ein Enumerationsliteral von ENUMXMLValidationMode[382] ist, welches festlegt, ob die Gültigkeit oder Wohlgeformtheit geprüft wird.

COM und .NET

Signatur: **`XMLValidationMode(ENUMXMLValidationMode[382] valMode)`**

Java

Signatur: **`public void setXMLValidationMode(ENUMXMLValidationMode[382] valMode)`**

6.4.1.5.2.22 XQueryFileName

Definiert die zu verwendende XQuery-Datei. Der bereitgestellte String muss eine absolute URL sein, die den Basispfad der zu verwendenden XQuery-Datei angibt.

COM und .NET

Signatur: **`XQueryFileName(string fileurl)`**

Java

Signatur: **`public void setXQueryFileName(string fileurl)`**

6.4.1.5.2.23 XQueryFromText

Liefert den Inhalt des zu verwendenden XQuery-Dokuments in Form eines Text-Strings.

COM und .NET

Signatur: `XQueryFromText(string xqtext)`

Java

Signatur: `public void setXQueryFromText(string xqtext)`

6.4.1.5.2.24 XQueryUpdateVersion

Definiert, welche XQuery Update-Version verwendet werden soll. Der Wert der Eigenschaft ist ein ENUMXQueryVersion[384] Literal.

COM und .NET

Signatur: `XQueryUpdateVersion(ENUMXQueryVersion[384] version)`

Java

Signatur: `public void setXQueryUpdateVersion(ENUMXQueryVersion[384] version)`

6.4.1.5.2.25 XSDVersion

Definiert die XML-Schema-Version, anhand welcher das XML-Dokument validiert wird. Der Wert ist ein Enumerationsliteral von ENUMXSDVersion[385].

COM und .NET

Signatur: `XSDVersion(ENUMXSDVersion[385] version)`

Java

Signatur: `public void setXSDVersion(ENUMXSDVersion[385] version)`

6.4.1.6 XSLT

Die **IXSLT/XSLT**-Schnittstelle/Klasse bietet Methoden zum Ausführen von XSLT-Transformationen und Validieren von Dokumenten im Zusammenhang mit XSLT. Sie können auch über externe Parameter Daten für die Transformation bereitstellen.

Beachten Sie, dass der Name der Schnittstelle in der COM/.NET API nicht derselbe ist wie der der Klasse in der Java API:

- In COM/.NET: **IXSLT**
- In Java: **XSLT**

6.4.1.6.1 Methoden

In diesem Abschnitt sind die Methoden der **IXSLT**-Schnittstelle (COM/.NET) und der **XSLT**-Klasse (Java) beschrieben.

6.4.1.6.1.1 AddExternalParameter

Fügt den Namen und Wert eines neuen externen Parameters hinzu. Jeder externe Parameter und sein Wert müssen in einem separaten Aufruf der Methode definiert werden. Die Parameter müssen im XSLT-Dokument deklariert sein. Da Parameterwerte XPath-Ausdrücke sind, müssen Parameterwerte, die Strings sind, in einfache Anführungszeichen gesetzt werden. Der Parameter `name` enthält den Namen des Parameters in Form eines QName als String. Der Parameter `value` enthält den Wert des Parameters als String.

COM und .NET

Signatur: **AddExternalParameter(**string **name,** string **value)**

Java

Signatur: **public** void **addExternalParameter(**string **name,** string **value)**

6.4.1.6.1.2 ClearExternalParameterList

Löscht die mit der Methode AddExternalParameter[365] erstellte Liste der externen Parameter.

COM und .NET

Signatur: **ClearExternalParameterList()**

Java

Signatur: `public void clearExternalParameterList()`

6.4.1.6.1.3 Execute

Führt die XSLT-Transformation anhand der in der `EngineVersion`[357] Eigenschaft genannten XSLT-Spezifikation aus und speichert das Ergebnis in der im Parameter `outputFile` genannten Ausgabedatei. Bei Auftreten eines Fehlers wird eine `RaptorXMLException`[333] ausgelöst. Mit Hilfe der `LastErrorMessage` Eigenschaft können Sie zusätzliche Informationen aufrufen.

COM und .NET

Signatur: `boolean Execute(string outputFile)`

Java

Signatur: `public boolean execute(string outputFile)`

6.4.1.6.1.4 ExecuteAndGetResultAsString

Führt die XSLT-Transformation anhand der in der Eigenschaft `EngineVersion`[368] genannten XSLT-Spezifikation aus und gibt das Ergebnis als String zurück. Mit dieser Methode werden keine zusätzlichen Ergebnisdateien wie z.B. Diagramme oder sekundäre Ergebnisse erzeugt. Sie enthält auch keine Binärdateiergebnisse wie z.B. `.docx` OOXML-Dateien. Falls zusätzliche Ausgabedateien benötigt werden, verwenden Sie die `Execute`[366]-Methode. Bei Auftreten eines Fehlers wird eine `RaptorXMLException`[333] ausgelöst. Mit Hilfe der `LastErrorMessage` Eigenschaft können Sie zusätzliche Informationen aufrufen.

COM und .NET

Signatur: `string ExecuteAndGetResultAsString()`

Java

Signatur: `public string executeAndGetResultAsString()`

6.4.1.6.1.5 ExecuteAndGetResultAsStringWithBaseOutputURI

Führt die XSLT-Transformation anhand der in der Eigenschaft `EngineVersion`[368] genannten XSLT-Spezifikation aus und gibt das Ergebnis unter dem durch die Basis-URI definierten Pfad als String zurück. Der Parameter

baseURI ist ein String, der eine URI liefert. Mit dieser Methode werden keine zusätzlichen Ergebnisdateien wie z.B. Diagramme oder sekundäre Ergebnisse erzeugt. Sie enthält auch keine Binärdateiergebnisse wie z.B. .docx OOXML-Dateien. Falls zusätzliche Ausgabedateien benötigt werden, verwenden Sie die Execute[366]-Methode. Bei Auftreten eines Fehlers wird eine RaptorXMLException[333] ausgelöst. Mit Hilfe der LastErrorMessage Eigenschaft können Sie zusätzliche Informationen aufrufen.

COM und .NET

Signatur: `string ExecuteAndGetResultAsStringWithBaseOutputURI(string baseURI)`

Java

Signatur: `public string ExecuteAndGetResultAsStringWithBaseOutputURI(string baseURI)`

6.4.1.6.1.6 IsValid

Gibt das Ergebnis der Validierung des XSLT-Dokuments, die anhand der in der Eigenschaftr EngineVersion[368] genannten XSLT-Spezifikation durchgeführt wurde, zurück. Das Ergebnis ist bei Erfolg `true`, bei Fehlschlag `false`. Bei Auftreten eines Fehlers wird eine RaptorXMLException[333] ausgelöst. Mit Hilfe der LastErrorMessage Methode können Sie zusätzliche Informationen aufrufen.

COM und .NET

Signatur: `boolean IsValid()`

Java

Signatur: `public boolean isValid()`

6.4.1.6.2 Eigenschaften

In diesem Abschnitt sind die Eigenschaften der **IXSLT**-Schnittstelle (COM/.NET) und der **XSLT**-Klasse (Java) beschrieben.

6.4.1.6.2.1 AdditionalOutputs

Gibt die zusätzlichen Ausgabedokumente des zuletzt ausgeführten Auftrags zurück.

COM und .NET

Signatur: `string AdditionalOutputs()`

Java

Signatur: `public string getAdditionalOutputs()`

6.4.1.6.2.2 ChartExtensionsEnabled

Aktiviert bzw. deaktiviert Altova-Diagrammerweiterungsfunktionen. Ein Wert `true` aktiviert Diagrammerweiterungen; `false` deaktiviert sie. Der Standardwert ist `true`.

COM und .NET

Signatur: `ChartExtensionsEnabled(boolean enable)`

Java

Signatur: `public void setChartExtensionsEnabled(boolean enable)`

6.4.1.6.2.3 DotNetExtensionsEnabled

Aktiviert oder deaktiviert .NET-Erweiterungsfunktionen. Ein Wert `true` aktiviert .NET-Erweiterungen; `false` deaktiviert sie. Der Standardwert ist `true`.

COM und .NET

Signatur: `DotNetExtensionsEnabled(boolean enable)`

Java

Signatur: `public void setDotNetExtensionsEnabled(boolean enable)`

6.4.1.6.2.4 EngineVersion

Definiert, welche XSLT-Version verwendet werden soll. Der Eigenschaftswert ist ein ENUMXSLTVersion 386 Literal.

COM und .NET

Signatur: `EngineVersion(ENUMXSLTVersion 386 version)`

Java

Signatur: `public void setEngineVersion(ENUMXSLTVersion version)`

6.4.1.6.2.5 IndentCharacters

Definiert den in der Ausgabe als Einrückung zu verwendenden Zeichenstring.

COM und .NET

Signatur: `IndentCharacters(string indentChars)`

Java

Signatur: `public void setIndentCharacters(string indentChars)`

6.4.1.6.2.6 InitialTemplateMode

Definiert den Anfangsmodus für die XSLT-Verarbeitung. Vorlagen mit einem Moduswert, der dem bereitgestellten String entspricht, werden verarbeitet.

COM und .NET

Signatur: `InitialTemplateMode(string mode)`

Java

Signatur: `public void setInitialTemplateMode(string mode)`

6.4.1.6.2.7 InputXMLFileName

Definiert den Pfad des zu verarbeitenden XML-Dokuments in Form einer URL. Der bereitgestellte String muss eine absolute URL sein, die den genauen Pfad der XML-Datei angibt.

COM und .NET

Signatur: `InputXMLFileName(string url)`

Java

Signatur: `public void setInputXMLFileName(string url)`

6.4.1.6.2.8 InputXMLFromText

Liefert den Inhalt des zu verarbeitenden XML-Dokuments. Der bereitgestellte String ist der Inhalt des zu verarbeitenden XML-Dokuments.

COM und .NET

Signatur: `InputXMLFromText(string xml)`

Java

Signatur: `public void setInputXMLFromText(string xml)`

6.4.1.6.2.9 JavaBarcodeExtensionLocation

Definiert den Pfad der Java-Barcode-Erweiterungsdatei. Nähere Informationen dazu finden Sie im Abschnitt Barcode-Erweiterungsfunktionen von Altova 526. Der bereitgestellte String muss eine absolute URL sein, die den Basispfad der zu verwendenden Datei angibt.

COM und .NET

Signatur: `JavaBarcodeExtensionLocation(string url)`

Java

Signatur: `public void setJavaBarcodeExtensionLocation(string url)`

6.4.1.6.2.10 JavaExtensionsEnabled

Aktiviert oder deaktiviert Java-Erweiterungsfunktionen. Der Wert `true` aktiviert Java-Erweiterungen; `false` deaktiviert sie. Der Standardwert ist `true`.

COM und .NET

Signatur: `JavaExtensionsEnabled(boolean enable)`

Java

Signatur: `public void setJavaExtensionsEnabled(boolean enable)`

6.4.1.6.2.11 LastErrorMessage

Ruft einen String auf, der die letzte Fehlermeldung aus dem RaptorXML-Prozessor darstellt, auf.

COM und .NET

Signatur: `string LastErrorMessage()`

Java

Signatur: `public string getLastErrorMessage()`

6.4.1.6.2.12 LoadXMLWithPSVI

Aktiviert die Validierung von XML-Dateien und generiert eine Post-Schema-Validierungsinfo für diese. Mit dem Wert `true` wird die XML-Validierung aktiviert und eine Post-Schema-Validierungsinfo für die XML-Dateien generiert; `false` deaktiviert die Validierung. Standardwert ist `true`.

COM und .NET

Signatur: `LoadXMLWithPSVI(boolean enable)`

Java

Signatur: `public void setLoadXMLWithPSVI(boolean enable)`

6.4.1.6.2.13 MainOutput

Gibt die Hauptausgabe des zuletzt ausgeführten Auftrags zurück.

COM und .NET

Signatur: `string MainOutput()`

Java

Signatur: `public string getMainOutput()`

6.4.1.6.2.14 NamedTemplateEntryPoint

Definiert den Namen der benannten Vorlage, die als Einstiegspunkt für die Transformation verwendet werden soll, als String.

COM und .NET

Signatur: `NamedTemplateEntryPoint(string template)`

Java

Signatur: `public void setNamedTemplateEntryPoint(string template)`

6.4.1.6.2.15 SchemaImports

Definiert, wie Schemaimporte auf Basis der Attributwerte der `xs:import` Elemente zu behandeln sind. Die Art der Behandlung wird durch das bereitgestellte `ENUMSchemaImports`-Literal definiert.

COM und .NET

Signatur: `SchemaImports(ENUMSchemaImports 378 importOption)`

Java

Signatur: `public void setSchemaImports(ENUMSchemaImports 378 importOption)`

6.4.1.6.2.16 SchemalocationHints

Definiert, welcher Mechanismus zum Auffinden des Schemas verwendet werden soll. Der Mechanismus wird durch das ausgewählte `ENUMLoadSchemalocation` Literal definiert.

COM und .NET

Signatur: `SchemalocationHints(ENUMLoadSchemalocation 377 hint)`

Java

Signatur: `public void setSchemalocationHints(ENUMLoadSchemalocation hint)`

6.4.1.6.2.17 SchemaMapping

Definiert, welches Mapping zum Auffinden des Schemas verwendet werden soll. Das Mapping wird durch das ausgewählte `ENUMSchemaMapping`-Literal definiert.

COM und .NET

Signatur: `SchemaMapping(ENUMSchemaMapping mappingOption)`

Java

Signatur: `public void setSchemaMapping(ENUMSchemaMapping mappingOption)`

6.4.1.6.2.18 StreamingSerialization

Aktiviert die Streaming-Serialisierung. Im Streaming-Modus werden möglichst wenige Daten im Arbeitsspeicher gehalten, wodurch die Verarbeitung beschleunigt wird. Der Wert `true` aktiviert die Streaming-Serialisierung; `false` deaktiviert sie.

COM und .NET

Signatur: `StreamingSerialization(boolean enable)`

Java

Signatur: `public void setStreamingSerialization(boolean enable)`

6.4.1.6.2.19 XincludeSupport

Aktiviert oder deaktiviert die Verwendung von `XInclude` Elementen. Der Wert `true` aktiviert die XInclude-Unterstützung; `false` deaktiviert sie. Der Standardwert ist `false`.

COM und .NET

Signatur: `XincludeSupport(boolean xinclude)`

Java

Signatur: `public void setXincludeSupport(boolean xinclude)`

6.4.1.6.2.20 XMLValidationErrorsAsWarnings

Aktiviert/Deaktiviert die Behandlung von XML-Validierungsfehlern als Warnungen. Erhält den Booleschen Wert `true` oder `false`.

COM und .NET

Signatur: `XMLValidationErrorsAsWarnings(boolean enable)`

Java

Signatur: `public void setXMLValidationErrorsAsWarnings(boolean enable)`

6.4.1.6.2.21 XMLValidationMode

Definiert den XML-Validierungsmodus, welcher ein Enumerationsliteral von ENUMXMLValidationMode[382] ist, welches festlegt, ob die Gültigkeit oder Wohlgeformtheit geprüft wird.

COM und .NET

Signatur: `XMLValidationMode(ENUMXMLValidationMode[382] valMode)`

Java

Signatur: `public void setXMLValidationMode(ENUMXMLValidationMode[382] valMode)`

6.4.1.6.2.22 XSDVersion

Definiert die XML-Schema-Version, anhand welcher das XML-Dokument validiert wird. Der Wert ist ein Enumerationsliteral von ENUMXSDVersion[385].

COM und .NET

Signatur: `XSDVersion(ENUMXSDVersion[385] version)`

Java

Signatur: `public void setXSDVersion(ENUMXSDVersion 385 version)`

6.4.1.6.2.23 XSLFileName

Definiert die für die Transformation zu verwendende XSLT-Datei. Der bereitgestellte String muss eine absolute URL sein, die den genauen Pfad zur zu verwendenden XSLT-Datei angibt.

COM und .NET

Signatur: `XSLFileName(string fileurl)`

Java

Signatur: `public void setXSLFileName(string fileurl)`

6.4.1.6.2.24 XSLFromText

Liefert den Inhalt des für die Transformation zu verwendenden XSLT-Dokuments als Textstring.

COM und .NET

Signatur: `XSLFromText(string xsltext)`

Java

Signatur: `public void setXSLFromText(string xsltext)`

6.4.2 Enumerationen

In diesem Abschnitt sind die Enumerationen der COM/.NET- und der Java Server API beschrieben. Die einzelnen Beschreibungen enthalten Links zu den Methoden oder Eigenschaften, in denen die Enumeration verwendet wird.

- ENUMAssessmentMode 376
- ENUMErrorFormat 376
- ENUMLoadSchemalocation 377
- ENUMSchemaImports 378
- ENUMSchemaMapping 379

- ENUMValidationType 380
- ENUMWellformedCheckType 382
- ENUMXMLValidationMode 382
- ENUMXQueryUpdatedXML 383
- ENUMXQueryVersion 384
- ENUMXSDVersion 385
- ENUMXSLTVersion 386

6.4.2.1 ENUMAssessmentMode

Definiert, ob der Validierungsmodus des XML-Validators auf strict oder lax gesetzt wird:

- **eAssessmentModeStrict**: Setzt den Schema-Validierungsmodus auf `Strict`. Dies ist der Standardwert.
- **eAssessmentModeLax**: Setzt den Schema-Validierungsmodus auf `Lax`.

COM und .NET

eAssessmentModeStrict	= 0
eAssessmentModeLax	= 1

Verwendet von

Schnittstelle	*Eigenschaft*
IXMLValidator 340	AssessmentMode 343

Java

```
public enum ENUMAssessmentMode {
       eAssessmentModeLax
       eAssessmentModeStrict   }
```

Verwendet von

Klasse	*Methode*
XMLValidator 340	setAssessmentMode 343

6.4.2.2 ENUMErrorFormat

Definiert das Format der Fehlerausgabe:

- **eFormatText**: Definiert das Fehlerausgabeformat als `Text`. Dies ist der Standardwert.
- **eFormatShortXML**: Definiert das Fehlerausgabeformat als `ShortXML`. Dieses Format ist eine Kurzform des Formats `LongXML`.

- **eFormatLongXML**: Definiert das Fehlerausgabeformat als `LongXML`. Dieses Format ist das ausführlichste Fehlerausgabeformat.

COM und .NET

eFormatText	= 0
eFormatShortXML	= 1
eFormatLongXML	= 2

Verwendet von

Schnittstelle	*Eigenschaft*
IServer 323	ErrorFormat 327

Java

```
public enum ENUMErrorFormat {
        eFormatText
        eFormatShortXML
        eFormatLongXML }
```

Verwendet von

Klasse	*Methode*
RaptorXMLFactory 323	setErrorFormat 327

6.4.2.3 ENUMLoadSchemalocation

Gibt an, wie der Schemapfad ermittelt werden soll. Die Auswahl basiert auf dem Schemapfadattribut des XML-Instanzdokuments. Dieses Attribut kann `xsi:schemaLocation` oder `xsi:noNamespaceSchemaLocation` sein.

- **eSHLoadBySchemalocation** verwendet die URL des Schemapfadattributs im XML- oder Instanzdokuments. Dieses Enumerationsliteral ist der **Standardwert**.
- **eSHLoadByNamespace** verwendet den Namespace-Teil von `xsi:schemaLocation` und im Fall von `xsi:noNamespaceSchemaLocation` einen leeren String und ermittelt das Schema über ein Katalogmapping.
- **eSHLoadCombiningBoth**: Wenn entweder die Namespace URL oder der Schemapfad ein Katalogmapping hat, so wird das Katalogmapping verwendet. Haben beide Katalogmappings, hängt es vom Wert des ENUMSchemaMapping 379-Parameters ab, welches Mapping verwendet wird. Wenn weder der Namespace noch der Schemapfad ein Katalogmapping hat, wird die Schemapfad-URL verwendet.
- **eSHLoadIgnore**: Die Attribute `xsi:schemaLocation` und `xsi:noNamespaceSchemaLocation` werden beide ignoriert.

COM und .NET

eSHLoadBySchemalocation	= 0
eSHLoadByNamespace	= 1
eSHLoadCombiningBoth	= 2
eSHLoadIgnore	= 3

Verwendet von

Schnittstelle	*Eigenschaft*
IXMLValidator 340	SchemalocationHints 350
IXSLT 365	SchemalocationHints 372

Java

```
public enum ENUMLoadSchemalocation {
        eSHLoadBySchemalocation
        eSHLoadByNamespace
        eSHLoadCombiningBoth
        eSHLoadIgnore    }
```

Verwendet von

Klasse	*Methode*
XMLValidator 340	setSchemalocationHints 350
XSLT 365	setSchemalocationHints 372

6.4.2.4 ENUMSchemaImports

Definiert das Verhalten der `xs:import`-Elemente des Schemas, von denen jedes ein optionales `namespace` Attribut und ein optionales `schemaLocation` Attribut hat.

- **`eSILoadBySchemalocation`** Das Schema wird unter Berücksichtigung von Katalogmappings anhand des Werts des `schemaLocation`-Attributs gesucht. Ist das `namespace`-Attribut vorhanden, wird der Namespace importiert (lizenziert).
- **`eSILoadPreferringSchemalocation`**: Wenn das `schemaLocation`-Attribut vorhanden ist, wird es verwendet, wobei Katalogmappings berücksichtigt werden. Ist kein `schemaLocation`-Attribut vorhanden, so wird der Wert des `namespace`-Attributs über ein Katalogmapping verwendet. Dieses Enumerationsliteral ist der **Standardwert**.
- **`eSILoadByNamespace`** Das Schema wird anhand des Werts des `namespace`-Attributs über ein Katalogmapping gesucht.
- **`eSILoadCombiningBoth`**: Wenn entweder die `namespace`-URL oder die `schemaLocation`-URL ein Katalogmapping hat, so wird das Katalogmapping verwendet. Wenn beide Attribute Katalogmappings

haben, hängt es vom ENUMSchemaMapping[379]-Parameter ab, welches Mapping verwendet wird. Wenn kein Katalogmapping vorhanden ist, wird die schemaLocation-URL verwendet.
- **eSILicenseNamespaceOnly**: Der Namespace wird importiert. Es wird kein Schema-Dokument importiert.

COM und .NET

eSILoadBySchemalocation	= 0
eSILoadPreferringSchemalocation	= 1
eSILoadByNamespace	= 2
eSICombiningBoth	= 3
eSILicenseNamespaceOnly	= 4

Verwendet von

Schnittstelle	*Eigenschaft*
IXMLValidator[340]	SchemaImports[350]
IXSLT[365]	SchemaImports[372]

Java

```
public enum ENUMSchemaImports {
        eSILoadBySchemalocation
        eSILoadPreferringSchemalocation
        eSILoadByNamespace
        eSILoadCombiningBoth
        eSILicenseNamespaceOnly   }
```

Verwendet von

Klasse	*Methode*
XMLValidator[340]	setSchemaImports[350]
XSLT[365]	setSchemaImports[372]

6.4.2.5 ENUMSchemaMapping

Definiert, welches der beiden Katalogmappings verwendet wird: Namespace- oder Schemapfad-URL. Diese Enumeration eignet sich, um ENUMLoadSchemalocation[377] und ENUMSchemaImports[378] eindeutig zu unterscheiden.

- **eSMPreferNamespace**: Wählen den Namespace aus.
- **eSMPreferSchemalocation**: Wählen den Schemapfad aus. Dies ist der Standardwert.

COM und .NET

eSMPreferSchemalocation	= 0
eSMPreferNamespace	= 1

Verwendet von

Schnittstelle	*Eigenschaft*
IXMLValidator 340	SchemaMapping 351
IXSLT 365	SchemaMapping 373

Java

```
public enum ENUMSchemaMapping {
       eSMPreferSchemalocation
       eSMPreferNamespace    }
```

Verwendet von

Klasse	*Methode*
IXMLValidator 340	setSchemaMapping 351
IXSLT 365	setSchemaMapping 373

6.4.2.6 ENUMValidationType

Gibt an, welche Validierung durchgeführt werden soll und definiert im Fall von XML-Dokumenten, ob die Validierung anhand einer DTD oder einer XSD durchgeführt werden soll.

- **eValidateAny**: Der Dokumenttyp (z.B. XML oder XSD) wird ermittelt und die Validierung wird automatisch für diesen Dokumenttyp ausgewählt.
- **eValidateXMLWithDTD**: Definiert die Validierung eines XML-Dokuments anhand einer DTD.
- **eValidateXMLWithXSD**: Definiert die Validierung eines XML-Dokuments anhand einer XSD (XML-Schema).
- **eValidateDTD**: Definiert die Validierung eines DTD-Dokuments.
- **eValidateXSD**: Definiert die Validierung eines XSD (W3C XML-Schema)-Dokuments.
- **eValidateJSON**: Definiert die Validierung eines JSON-Instanzdokuments.
- **eValidateJSONSchema**: Definiert die Validierung eines JSON-Schemadokuments anhand von JSON Schema v4.
- **eValidateAvro**: Definiert die Validierung einer Avro-Binärdatei. Die Avro-Daten in der Binärdatei werden anhand des in der Binärdatei enthaltenen Schemas validiert.
- **eValidateAvroSchema**: Definiert die Validierung eines Avro-Schemas anhand der Avro-Schema-Spezifikation.

- **eValidateAvroJSON**: Definiert die Validierung einer JSON-serialisierten Avro-Datendatei anhand eines Avro-Schemas.

COM und .NET

eValidateAny	= 0
eValidateXMLWithDTD	= 1
eValidateXMLWithXSD	= 2
eValidateDTD	= 3
eValidateXSD	= 4
eValidateJSON	= 5
eValidateJSONSchema	= 6
eValidateAvro	= 7
eValidateAvroSchema	= 8
eValidateAvroJSON	= 9

Verwendet von

Schnittstelle	*Methode*
IXMLValidator 340	isValid 342

Java

```
public enum ENUMValidationType {
       eValidateAny
       eValidateXMLWithDTD
       eValidateXMLWithXSD
       eValidateDTD
       eValidateXSD
       eValidateJSON
       eValidateJSONSchema
       eValidateAvro
       eValidateAvroSchema
       eValidateAvroJSON }
```

Verwendet von

Klasse	*Methode*
XMLValidator 340	isValid 342

6.4.2.7 ENUMWellformedCheckType

Definiert, welche Art der Wohlgeformtheitsprüfung durchgeführt werden soll (für XML, DTD oder JSON).

- **eWellformedAny**: Der Dokumenttyp wird ermittelt und die Art der Überprüfung wird automatisch ausgewählt.
- **eWellformedXML**: Überprüft ein XML-Dokument auf Wohlgeformtheit.
- **eWellformedDTD**: Überprüft ein DTD-Dokument auf Wohlgeformtheit.
- **eWellformedJSON**: Überprüft ein JSON-Dokument auf Wohlgeformtheit.

COM und .NET

eWellFormedAny	= 0
eWellFormedXML	= 1
eWellFormedDTD	= 2
eWellFormedJSON	= 3

Verwendet von

Schnittstelle	*Methode*
IXMLValidator 340	isWellFormed 342

Java

```
public enum ENUMWellformedCheckType {
        eWellformedAny
        eWellformedXML
        eWellformedDTD
        eWellformedJSON    }
```

Verwendet von

Klasse	*Methode*
XMLValidator 340	isWellFormed 342

6.4.2.8 ENUMXMLValidationMode

Definiert, welche Art von XML-Validierung durchgeführt werden soll (Validierung oder Wohlgeformtheitsprüfung).

- **eProcessingModeWF**: Setzt den XML-Verarbeitungsmodus auf `Wellformed`. Dies ist der Standardwert.
- **eProcessingModeValid**: Setzt den XML-Verarbeitungsmodus auf `validation`.
- **eProcessingModeID**: Intern. Soll nicht verwendet werden.

COM und .NET

eXMLValidationModeWF	= 0
eXMLValidationModeID	= 1
eXMLValidationModeValid	= 2

Verwendet von

Schnittstelle	*Eigenschaft*
IXMLValidator 340	XMLValidationMode 352
IXQuery 353	XMLValidationMode 363
IXSLT 365	XMLValidationMode 374

Java

```
public enum ENUMXMLValidationMode {
       eProcessingModeValid
       eProcessingModeWF
       eProcessingModeID  }
```

Verwendet von

Klasse	*Methode*
XMLValidator 340	setXMLValidationMode 352
XQuery 353	setXMLValidationMode 363
XSLT 365	setXMLValidationMode 374

6.4.2.9 ENUMXQueryUpdatedXML

Definiert, wie XQuery Updates behandelt werden sollen.

- **eUpdatedDiscard**: Aktualisierungen werden verworfen und nicht in eine Datei geschrieben.
- **eUpdatedWriteback**: Aktualisierungen werden in die mit (set)InputXMLFileName 358 definierte XML-Input-Datei geschrieben.
- **eUpdatedAsMainResult**: Aktualisierungen werden in die Datei geschrieben, die durch den Parameter outputFile von ExecuteUpdate 334 definiert ist.

COM und .NET

eUpdatedDiscard	= 1
eUpdatedWriteback	= 2

eUpdatedAsMainResult	= 3

Verwendet von

Schnittstelle	*Eigenschaft*
IXQuery 353	UpdatedXMLWriteMode 362

Java

```
public enum ENUMXQueryUpdatedXML {
        eUpdatedDiscard
        eUpdatedWriteback
        eeUpdatedAsMainResult }
```

Verwendet von

Klasse	*Methode*
XQuery 353	setUpdatedXMLWriteMode 362

6.4.2.10 ENUMXQueryVersion

Definiert, welche XQuery-Version für die Verarbeitung verwendet werden soll (Ausführung oder Validierung).

- **eXQVersion10**: Definiert XQuery 1.0 als die zu verwendende XQuery-Version.
- **eXQVersion30**: Definiert XQuery 3.0 als die zu verwendende XQuery-Version. Dies ist der Standardwert.
- **eXQVersion31**: Definiert XQuery 3.1 als die zu verwendende XQuery-Version.

Anmerkung: Die Java-Eumerationsliterale haben einen anderen Namen als die COM/.NET-Literale. *Siehe unten.*

COM und .NET

eXQVersion10	= 1
eXQVersion30	= 3
eXQVersion31	= 31

Verwendet von

Schnittstelle	*Eigenschaft*
IXQuery 353	EngineVersion 357

Java

```
public enum ENUMXQueryVersion {
```

```
        eVersion10
        eVersion30
        eVersion31 }
```

Verwendet von

Klasse	*Methode*
XQuery 353	setEngineVersion 357

6.4.2.11 ENUMXSDVersion

Definiert, welche XML-Schema-Version für die Validierung verwendet werden soll.

- **eXSDVersionAuto**: Die XSD-Version wird automatisch anhand des `vc:minVersion`-Attributs des XSD-Dokuments ermittelt. Wenn das Attribut den Wert `1.1` hat, wird das Dokument als XSD 1.1 erkannt. Wenn das Attribut einen anderen Wert hat oder fehlt, wird das Dokument als XSD 1.0 erkannt.
- **eXSDVersion10**: Setzt die für die Dokumentvalidierung zu verwendende XML-Schema-Version auf XML-Schema 1.0.
- **eXSDVersion11**: Setzt die für die Dokumentvalidierung zu verwendende XML-Schema-Version auf XML-Schema 1.1.

COM und .NET

eXSDVersionAuto	= 0
eXSDVersion10	= 1
eXSDVersion11	= 2

Verwendet von

Schnittstelle	*Eigenschaft*
IXMLValidator 340	XSDVersion 352
IXQuery 353	XSDVersion 364
IXSLT 365	XSDVersion 374

Java

```
public enum ENUMXSDVersion {
        eXSDVersionAuto
        eXSDVersion10
        eXSDVersion11    }
```

Verwendet von

Klasse	*Methode*

XMLValidator 340	setXSDVersion 352
XQuery 353	setXSDVersion 364
XSLT 365	setXSDVersion 374

6.4.2.12 ENUMXSLTVersion

Definiert, welche XSLT-Version für die Verarbeitung (Validierung oder XSLT-Transformation) verwendet werden soll.

- **eVersion10**: Definiert XSLT 1.0 als die zu verwendende XSLT-Version.
- **eVersion20**: Definiert XSLT 2.0 als die zu verwendende XSLT-Version.
- **eVersion30**: Definiert XSLT 3.0 als die zu verwendende XSLT-Version.

COM und .NET

eVersion10	= 1
eVersion20	= 2
eVersion30	= 3

Verwendet von

Schnittstelle	*Eigenschaft*
IXSLT 365	EngineVersion 368

Java

```
public enum ENUMXSLTVersion {
       eVersion10
       eVersion20
       eVersion30 }
```

Verwendet von

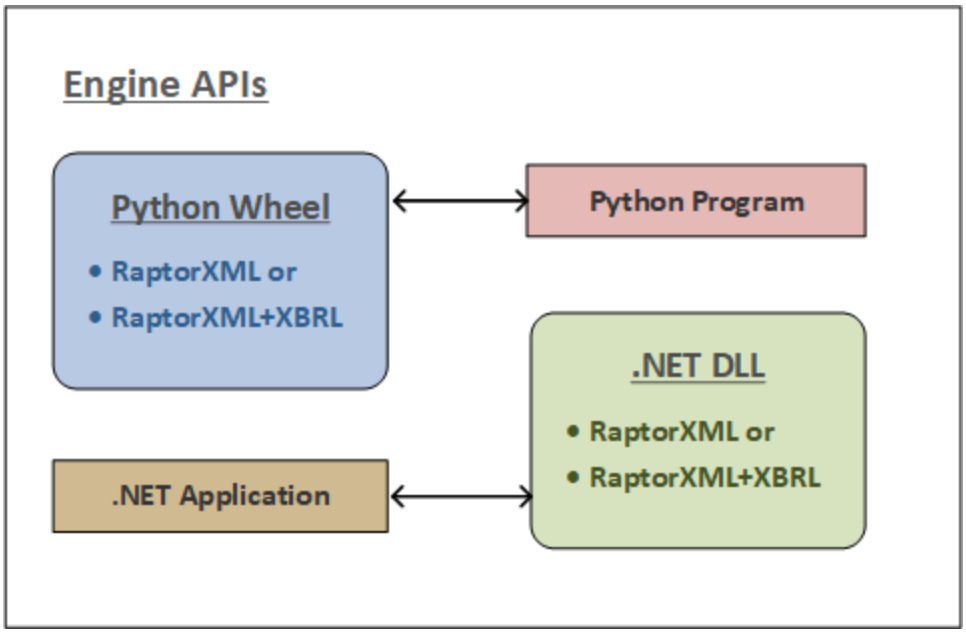
Klasse	*Methode*
XSLT 365	setEngineVersion 368

7 Prozessor APIs: Python und .NET

RaptorXML Server stellt zwei Prozessor APIs zur Verfügung:

- eine Python Wheel-Datei (`.whl`), welche die Python-Prozessor API: `raptorxml<versiondetails>.whl` ist.
- eine .NET DLL-Datei (`.dll`), welche die .NET-Prozessor API: `raptorxmlapi.dll` ist.

Diese beiden Prozessor APIs stellen die RaptorXML Server-Funktionalitäten in Form separater Pakete, die eigenständig und unabhängig von RaptorXML Server sind, zur Verfügung (*siehe Abbildung unten*). Jedes Paket muss auf dem Rechner des Benutzers installiert werden, bevor es als Python-Modul importiert oder in eine benutzerdefinierte .NET-Applikation integriert werden kann. Da die gesamte Verarbeitung lokal auf dem Rechner des Benutzers erfolgt, bieten die Python- und die .NET-Prozessor API detaillierten Zugriff auf die Datenmodelle aller gültigen XML- und XBRL-Instanzen, XSD-Schemas und XBRL-Taxonomien. Über die APIs stehen zahlreiche Methoden zur Verfügung, um über den Inhalt von XBRL-Instanzen zu iterieren oder mit einigen Codezeilen bestimmte Informationen aus XBRL-Taxonomien abzurufen.



Beachten Sie die folgenden Punkte zu den Prozessor APIs:

- Nach Installation von RaptorXML Server befinden sich beide Prozessor APIs im Ordner `bin` des RaptorXML Server-Installationsordners.
- Die Prozessor APIs bieten dank flexiblerer Objekte in ihren APIs zusätzliche komplexe Verarbeitungsmöglichkeiten.
- Um eine Prozessor API verwenden zu können, muss auf dem Rechner, auf dem das Python-Programm oder die .NET-Applikation ausgeführt wird, eine lizenzierte Version von RaptorXML Server installiert sein (siehe Verwendung unten).

Verwendung

Folgendermaßen können Sie ein Python-Programm oder eine .NET-Applikation erstellen:

Python-Programm
Ein Python-Programm kann RaptorXML-Funktionalitäten mit Hilfe von Python API-Objekten[391] aufrufen (siehe hier[391]). Bei Ausführung des Python-Programms wird die RaptorXML-Bibliothek, die bei der Installation des Python Wheel in Ihrer Python-Umgebung installiert wurde, verwendet. Beachten Sie, dass das Python Wheel **nur** mit der Python-Version 3.11.8 kompatibel ist.

.NET-Applikation
Eine .NET-Applikation kann RaptorXML-Funktionalitäten mit Hilfe von .NET API-Objekten[401] aufrufen (siehe hier[401]). Bei Ausführung der .NET-Applikation wird der in der .NET API DLL enthaltene RaptorXML verwendet.

Lizenzierung

Um eine Prozessor API verwenden zu können, muss auf dem Rechner, auf dem das Python-Programm oder die .NET-Applikation ausgeführt wird, eine lizenzierte Version von RaptorXML Server installiert sein. Nähere Informationen dazu finden Sie im Abschnitt Lizenzierung[389].

7.1 Lizenzierung

Damit ein API-Paket auf einem Client-Rechner ausgeführt werden kann, muss dieser Rechner als RaptorXML Server-Client lizenziert worden sein. Die Lizenzierung erfolgt in zwei Schritten:

1. Registrieren des Rechners als RaptorXML Server-Client auf Altova LicenseServer
2. Zuweisen einer RaptorXML Server-Lizenz von LicenseServer an diesen Rechner.

Wenn Sie planen, das API-Paket von einem bestimmten Rechner aus zu verwenden, kann es zu zwei Situationen kommen:

- Wenn auf dem Client-Rechner bereits eine lizenzierte Version von RaptorXML Server ausgeführt wird, kann das API-Paket ausgeführt werden, ohne dass Sie zusätzliche Schritte setzen müssen, da die Ausführung von RaptorXML Server auf diesem Rechner bereits lizenziert wurde. Folglich wird auch die Verwendung des API-Pakets auf diesem Rechner von der RaptorXML Server auf diesem Rechner zugewiesenen Lizenz bereits abgedeckt.
- Wenn RaptorXML Server auf dem Client-Rechner nicht installiert ist und Sie RaptorXML Server auf diesem Rechner aus irgendeinem Grund nicht installieren möchten: Sie können den Rechner in diesem Fall dennoch als RaptorXML Server-Client registrieren und ihm eine RaptorXML Server-Lizenz zuweisen. Eine Anleitung dazu finden Sie im Folgenden.

Um einen Rechner (auf dem RaptorXML Server nicht installiert ist) als RaptorXML Server-Client zu registrieren, verwenden Sie die Befehlszeilenapplikation **`registerlicense.exe`** aus dem Ordner **`bin`** der Applikation:

Windows	`Programme\Altova\RaptorXMLServer2025\bin`
Linux	`/opt/Altova/RaptorXMLServer2025/bin`
Mac	`/usr/local/Altova/RaptorXMLServer2025/bin`

Führen Sie in der Befehlszeile den folgenden Befehl aus:

```
registerlicense <LicenseServer>
```

wobei **`<LicenseServer>`** für die IP-Adresse oder den Host-Namen des LicenseServer-Rechners steht.

Mit diesem Befehl wird der Rechner als RaptorXML Server-Client auf Altova LicenseServer registriert. Informationen darüber, wie Sie einem Rechner eine RaptorXML Server Lizenz zuweisen und nähere Informationen zur Lizenzierung finden Sie in der Dokumentation zu Altova LicenseServer.

Bereitstellung unter Linux

Um die **`registerlicense`**-Applikation mit Ihrem Python Wheel-Paket bereitzustellen, müssen sich die unten aufgelisteten gemeinsamen Bibliotheken in einem gleichrangigen `lib`-Verzeichnis befinden. Die gemeinsamen Bibliotheken können aus Ihrem Raptor-Installationsordner kopiert werden:

```
/opt/Altova/RaptorXMLServerRaptorXMLServer2025/lib
```

- `libcrypto.so.1.0.0`
- `libssl.so.1.0.0`
- `libstdc++.so.6`

- `libtbb.so.2`

7.2 Python API

Über die Python-API von RaptorXML können Daten in XML-Dokumenten und XML-Schema-Dokumenten in Python Skripts aufgerufen und verarbeitet werden. Dies sind einige typische Beispiele für die Verwendung der Python API:

- Implementierung von benutzerdefinierten Validierungsregeln und Fehlermeldungen
- Export von Inhalt aus XML-Dokumenten in eine Datenbank
- Export von Inhalt aus XML-Dokumenten in benutzerdefinierte Datenformate
- interaktive Navigation und Abfrage des Datenmodells von XML-Dokumenten in einer Python Shell oder einem Jupyter Notebook (http://jupyter.org/)

Die Python-APIs

Die Python-APIs (für XML und XSD) bieten Zugriff auf die Metainformationen, Strukturinformationen und Daten in XML- und XSD-Dokumenten. Somit können Python-Skripts erstellt werden, die über die APIs auf Dokumentinformationen zugreifen und diese verarbeiten. So kann z.B. ein Python-Skript an RaptorXML Server übergeben werden, das Daten aus einem XML-Dokument in eine Datenbank oder eine CSV-Datei schreibt.

Beispielskripts für die Pyhton APIs von Raptor finden Sie unter: https://github.com/altova

Die Python-APIs sind in den folgenden Abschnitten beschrieben:

- Python API v1 Reference
- Python API v2 Reference

Anmerkung: Die **Python API v1** von Raptor ist **veraltet**. Verwenden Sie stattdessen bitte Python API v2.

RaptorXML Server Paket für Python

Sie finden in Ihrer Installation von RaptorXML Server auch ein Python-Paket im Wheel-Formmat. Mit Hilfe des `pip`-Befehls von Python können Sie dieses Paket als Modul Ihrer Python-Installation installieren. Nach Installation des RaptorXML-Moduls können Sie die Funktionen des Moduls in Ihrem Code verwenden. Dadurch können Sie die RaptorXML-Funktionalitäten ganz einfach in jedem von Ihnen geschriebenen Python-Programm zusammen mit anderen Drittanbieter-Python-Bibliotheken wie z.B. Grafik-Bibliotheken nutzen.

Informationen zur Verwendung des Python-Pakets von RaptorXML Server finden Sie im Abschnitt RaptorXML Server als Python-Paket 394.

Anmerkung: Das Python Wheel in Versionen ab v2024r2 ist mit Python Versionen ab 3.11.8 kompatibel.

Python-Skripts

Ein vom Benutzer erstelltes Python-Skript wird mit dem Parameter `--script` einer Reihe von Befehlen wie den folgenden übergeben:

- `valxml-withxsd (xsi)` 65
- `valxsd (xsd)` 77

Diese Befehle, die Python-Skripts aufrufen, können sowohl über die Befehlszeilenschnittstelle (CLI)[58] als auch über die HTTP-Schnittstelle[268] verwendet werden. Die Verwendung von Python-Skripts mit den Python-APIs von RaptorXML Server ist unter https://github.com/altova beschrieben.

Python-Skripts sicher machen

Wenn ein Python-Skript in einem Befehl über HTTP an RaptorXML Server adressiert ist, funktioniert das Skript nur, wenn es sich im vertrauenswürdigen Verzeichnis[274] befindet. Das Skript wird vom vertrauenswürdigen Verzeichnis aus ausgeführt. Wenn Sie ein Python-Skript aus einem anderen Verzeichnis definieren, wird ein Fehler ausgegeben. Das vertrauenswürdige Verzeichnis wird in der `server.script-root-dir`[273] Einstellung der Serverkonfigurationsdatei[272] definiert. Wenn Sie Python-Skripts verwenden möchten, **muss** ein vertrauenswürdiges Verzeichnis definiert werden. Stellen Sie sicher, dass alle Python-Skripts, die verwendet werden sollen, in diesem Verzeichnis gespeichert werden.

Zwar werden alle vom Server für HTTP-Auftragsanforderungen generierten Ausgabedateien in das Auftragsausgabeverzeichnis[274] (ein Unterverzeichnis von `output-root-directory`[274]) geschrieben, doch gilt diese Einschränkung nicht für Python-Skripts, die in jeden Ordner geschrieben werden können. Der Server-Administrator muss die Python-Skripts im vertrauenswürdigen Verzeichnis[274] auf potentielle Schwachstellen überprüfen.

7.2.1 Python API-Versionen

RaptorXML Server unterstützt mehrere Python API-Versionen. Alle früheren Python API-Versionen werden auch von der aktuellen Version von RaptorXML Server unterstützt. Die Python API-Version wird vom Befehlszeilen-Flag **`--script-api-version=MAJOR_VERSION`** ausgewählt. Die Standardeinstellung des Arguments `MAJOR_VERSION` ist immer die aktuelle Version. Wenn nicht kompatible Änderungen oder größere Verbesserungen erfolgt sind, wird eine neue RaptorXML Server Python API `MAJOR_VERSION` verwendet. Benutzer, die die API verwenden, müssen Ihre vorhandenen Skripts nicht aktualisieren, wenn eine neue Hauptversion herauskommt.

Es wird empfohlen,

- dass Sie das Flag `--script-api-version=MAJOR_VERSION` verwenden, um Utility Skripts aus der RaptorXML Server Befehlszeile (oder Web API) aufzurufen. Damit stellen Sie sicher, dass Ihre Skripts nach RaptorXML Server Aktualisierungen weiterhin funktionieren, selbst wenn eine neue API `MAJOR_VERSION` herausgekommen ist.
- dass Sie für neue Projekte die neueste API-Version verwenden, auch wenn frühere Versionen in zukünftigen RaptorXML Server Versionen weiterhin unterstützt werden.

Die unten aufgelisteten Python API-Versionen stehen derzeit zur Verfügung. Sie finden die Dokumentation zu den verschiedenen APIs online unter den weiter unten angegebenen Adressen.

Beispieldateien

Beispiele für Skripts, in denen die Python API von Raptor verwendet wird, finden Sie unter https://github.com/altova.

Python API Version 1

Wird ab RaptorXML Server v2014 verwendet.

Befehlszeilen-Flag: **`--script-api-version=1`**
Dokumentation: Python API Version 1 Reference

Dies ist die Original-RaptorXML Server-Python API. Sie bietet Unterstützung zum Aufrufen des internen Modells von RaptorXML Server für:

- XML 1.0 und XML 1.1 (API-Modul **`xml`**)
- XMLSchema 1.0 und XMLSchema 1.1 (API-Modul **`xsd`**)
- XBRL 2.1 (API-Modul **`xbrl`**)

Die API kann über in einer Python-Skript-Datei implementierte Callback-Funktionen verwendet werden.

- `on_xsi_valid`
- `on_xsd_valid`
- `on_dts_valid`
- `on_xbrl_valid`

Mit der Option `--script` wird in der Befehlszeile ein Skript definiert. Die Callback-Funktionen werden nur aufgerufen, wenn die Validierung erfolgreich ist. Nähere Informationen zu Callback-Funktionen und der API finden Sie in der Referenz zur RaptorXML Server Python API Version 1.

Anmerkung: Die **Python API v1** von Raptor ist veraltet. Verwenden Sie stattdessen bitte Python API v2.

Python API Version 2

Wird ab RaptorXML Server v2015r3 verwendet. Die neueste API-Version ist `2.11.0`.

Befehlszeilen-Flag	Release
`--script-api-version=2`	*v 2015r3*
`--script-api-version=2,1`	*v 2015r4*
`--script-api-version=2,2`	*v 2016*
`--script-api-version=2,3`	*v 2016r2*
`--script-api-version=2,4`	*v 2017*
`--script-api-version=2.4.1`	*v 2018*
`--script-api-version=2.5.0`	*v 2018r2*
`--script-api-version=2.6.0`	*v 2019*
`--script-api-version=2.7.0`	*v2019r3*
`--script-api-version=2.8.0`	*v2020*
`--script-api-version=2.8.1`	*v2020r2*
`--script-api-version=2.8.2`	*v2021*

--script-api-version=2.8.3	*v2021r2*
--script-api-version=2.8.4	*v2022r2*
--script-api-version=2.8.5	*v2023r2sp1*
--script-api-version=2.8.6	*v2024*
--script-api-version=2.9.0	*v2024r2*
--script-api-version=2.10.0	*v2025*
--script-api-version=2.11.0	*v2025r2*
Dokumentation: Python API Version 2 Reference	

Diese API-Version enthält über 300 neue Klassen. Die Module aus der RaptorXML Server Python API Version 1 sind so gegliedert, dass häufig verwendete Informationen (z.B. PSVI-Daten) einfacher aufgerufen und miteinander in Zusammenhang stehende APIs logisch zusammen gruppiert werden (z.B. `xbrl.taxonomy`, `xbrl.formula`, `xbrl.table`). In dieser Version werden die Callback-Funktionen nicht nur bei erfolgreicher Validierung aufgerufen, sondern auch, wenn die Validierung fehlschlägt. Dies sehen Sie daran, dass der Name der Callback-Funktion folgendermaßen geändert wird:

- `on_xsi_finished`
- `on_xsd_finished`
- `on_dts_finished`
- `on_xbrl_finished`

Um die Verwendung von Modulen zu ermöglichen, unterstützt RaptorXML Server nun mehrere `--script` Optionen. Die in diesen Python Skript-Dateien implementierten Callbacks werden in der in der Befehlszeile angegebenen Reihenfolge ausgeführt.

7.2.2 RaptorXML Server als Python-Paket

Ab RaptorXML Server 2024 steht die Python API als natives Python Wheel-Paket für **Python 3.11.8** zur Verfügung. Das Python Wheel-Paket kann als Erweiterungsmodul in der Python 3.11.8 Distribution Ihrer Wahl (z.B. von python.org) installiert werden. Einige Python 3 Distributions (z.B. von jupyter.org, anaconda.org und SciPy.org) enthalten eine breite Palette an Erweiterungsmodulen für große Datenmengen, mathematische, wissenschaftliche, technische und grafische Anwendungen. Diese Module stehen RaptorXML Server nun zur Verfügung, ohne dass diese speziell für RaptorXML Server erstellt werden müssen. Das Wheel-Paket funktioniert auf dieselbe Art wie die mit RaptorXML Server inkludierte **`RaptorXMLXBRL-python.exe`** Applikation.

Anmerkung: Das Python Wheel-Paket ist ein natives Python 3.11.8-Erweiterungsmodul und mit der Python-Version 3.11.8 kompatibel.

Anmerkung: Das Python Wheel-Paket enthält die Python API v1 nicht.

Anmerkung: Stellen Sie bei einer Aktualisierung von RaptorXML Server sicher, dass Sie auch das Python Wheel-Paket in Ihrer Python-Umgebung aktualisieren.

In den nachstehenden Abschnitten finden Sie Informationen zur Installation des RaptorXML Server-Pakets:

Informationen zur Verwendung der Python API von RaptorXML Server finden Sie in der Python API-Referenz sowie den Beispielen 392. Beispiel-Skripts zur Verwendung der Python API von Raptor finden Sie auch unter https://github.com/altova.

Name der Wheel-Datei

Wheel-Dateien werden nach dem folgenden Muster benannt:

```
raptorxmlserver-{version}(-{build tag})?-{python tag}-{abi tag}-{platform tag}.whl
```

Beispiel:
```
raptorxmlserver-2.10.0-cp35-cp35m-win_amd64.whl
```

Pfad der Wheel-Datei

Im Installationspaket von RaptorXML Server ist eine Wheel-Datei enthalten. Sie befindet sich im **bin**-Ordner Ihrer Applikation:

Windows	`Programme\Altova\RaptorXMLServer2025\bin`
Linux	`/opt/Altova/RaptorXMLServer2025/bin`
Mac	`/usr/local/Altova/RaptorXMLServer2025/bin`

Installation einer Wheel-Datei mit pip

Um das RaptorXML Server-Paket als Python-Modul zu installieren, verwenden Sie den **pip**-Befehl:

```
pip install <wheel-file>.whl
python -m pip install <wheel-file>.whl
```

Wenn Sie Python 3.11.8 oder höher von python.org installiert haben, so ist `pip` bereits auf Ihrem Rechner installiert. Andernfalls müssen Sie zuerst **pip** installieren. Nähere Informationen dazu finden Sie unter https://docs.python.org/3/installing/.

Behebung von Fehlern bei der Installation

Falls Sie ältere Versionen des Python Interpreters verwenden, müssen Sie Ihre Installation eventuell anpassen, damit unter Windows die neuesten **vcruntime**-Bibliotheken oder unter Unix die C++-Standardbibliotheken verwendet werden. Diese Bibliotheken werden mit RaptorXML Server bereitgestellt und können wie unten beschrieben verwendet werden.

Windows
Falls die **vcruntime140_1.dll** fehlt, kopieren Sie sie aus dem Ordner `Programme\Altova\RaptorXMLServer2025\bin` in den Python-Installationsordner (den Ordner, der die

python.exe-Datei enthält). (D.h. der Python Interpreter muss wissen, wo sich die DLLs oder die gemeinsam verwendeten Bibliotheken befinden).

Linux
Wenn die C++-Bibliothek Ihres Systems veraltet ist, muss Ihr Python Interpreter wissen, wo er die neuere, von dem mit RaptorXML Server bereitgestellten RaptorXML Server-Python-Paket verwendete C++-Bibliothek findet. Verwenden Sie dazu **$LD_LIBRARY_PATH**, um auf die neuere Bibliothek im RaptorXML Server-Ordner zu verweisen: `$ export LD_LIBRARY_PATH=/opt/Altova/RaptorXMLServer2025/lib`.

macOS
Wenn die C++-Bibliothek Ihres Systems veraltet ist, muss Ihr Python Interpreter wissen, wo er die neuere, von dem mit RaptorXML Server bereitgestellten RaptorXML Server-Python-Paket verwendete C++-Bibliothek findet. Verwenden Sie dazu **$DYLD_LIBRARY_PATH**, um auf die neuere Bibliothek im RaptorXML Server-Ordner zu verweisen: `$ export DYLD_LIBRARY_PATH=/usr/local/Altova/RaptorXMLServer2025/lib`.

Die Root-Katalogdatei

Das RaptorXML-Modul Für Python muss in der Lage sein, **RootCatalog.xml**, die in Ihrem RaptorXML Server-Installationsordner gespeicherte Root-Katalogdatei, zu finden, damit das RaptorXML-Modul die verschiedenen Ressourcen wie Schemas und andere Spezifikationen, die das Modul zur Ausführung von Funktionen wie Validierung und Transformation referenziert, anhand des Katalogs findet. Wenn der Pfad der Katalogdatei nach der Installation von RaptorXML Server nicht geändert wurde, findet das RaptorXML-Modul die Datei **RootCatalog.xml** automatisch.

Wenn Sie Ihre RaptorXML Server-Umgebung verschieben oder ändern oder wenn Sie **RootCatalog.xml** aus dem ursprünglichen Installationsordner verschieben, können Sie den Pfad zur Katalogdatei mit Hilfe von Umgebungsvariablen und der JSON-Konfigurationsdatei des RaptorXML-Moduls[397] definieren. Die verschiedenen Methoden, wie Sie dies tun können, sehen Sie in der Liste unten. Mit dem RaptorXML-Modul wird der Pfad zur **RootCatalog.xml** durch Überprüfung der folgenden Ressourcen in der angegebenen Reihenfolge ermittelt.

1	Umgebungsvariable `ALTOVA_RAPTORXML_PYTHON_CATALOGPATH`	Wird mit einem Wert erstellt, der der Pfad zu **RootCatalog.xml** ist
2	HKLM Registry: `SOFTWARE\Altova\RaptorXMLServer\Installation_v2025_x64\Setup\CatalogPath`	Der Registrierungsschlüssel wird vom RaptorXML Server Installer hinzugefügt. Der Wert ist der Pfad zu **RootCatalog.xml**. *Nur Windows*
3	Pfad: `/opt/Altova/RaptorXMLServer2025/etc/RootCatalog.xml`	*Nur Linux*
4	Pfad: `/usr/local/Altova/RaptorXMLServer2025/etc/RootCatalog.xml`	*Nur Mac*
5	Umgebungsvariable `ALTOVA_RAPTORXML_PYTHON_CONFIG`	Wird mit einem Wert erstellt, der der Pfad zur JSON-Konfigurationsdatei[397] ist.
6	Pfad: `.altova/`**raptorxml-python.config**	Die JSON-Konfigurationsdatei[397] im aktuellen Arbeitsverzeichnis

7	Pfad: `~/.config/altova/raptorxml-python.config`	Die JSON-Konfigurationsdatei[397] im Startverzeichnis des Benutzers
8	Pfad: `/etc/altova/altova/raptorxml-python.config`	Die JSON-Konfigurationsdatei[397]. *Nur Linux und Mac*

Die JSON-Konfigurationsdatei

Sie können eine JSON-Konfigurationsdatei für das RaptorXMLServer-Modul erstellen. Diese Datei wird von den Optionen 5 bis 8 in der obigen Tabelle verwendet, um die Root-Katalogdatei[396] zu finden. Die JSON-Konfigurationsdatei muss eine Zuordnung mit dem Schlüssel `"CatalogPath"` enthalten, der einen Wert hat, der dem Pfad zur Root-Katalogdatei[396] entspricht.

Codefragment der JSON-Konfigurationsdatei

```
{
    "CatalogPath": "/path/to/RootCatalog.xml"
}
```

7.2.3 Debuggen von serverseitigen Python Scripts

Die meisten der Debugging-Funktionen - mit Ausnahme von Server-spezifischen Callbacks - können in einem Standard-Pyhton-Interpreter oder einer (virtuellen) Umgebung verwendet werden, nachdem das RaptorXML Server-Modul mittels `pip` installiert wurde:

```
pip install –upgrade "/path/to/RaptorXML/application-folder/bin/raptorxml-version-cp37-cp37m-winversion.whl"
```

After installing the wheel, you should be able to use any Python IDE to debug a script. You could try to extract the main functionality into a separate function which takes an instance object. This can then be called (i) by the RaptorXML Server callbacks, or (ii) by directly executing the script with a Python interpreter.

```
from altova_api.v2 import xml, xsd, xbrl

def main(instance):
    # Here goes the application specific logic

# Main entry point, will be called by RaptorXML after the XML instance validation job has finished
def on_xsi_finished(job, instance):
    # instance object will be None if XML Schema validation was not successful
    if instance:
        main(instance)

# Main entry point, will be called by RaptorXML after the XBRL instance validation job has finished
def on_xbrl_finished(job, instance):
    # instance object will be None if XBRL 2.1 validation was not successful
    if instance:
```

```
        main(instance)

if __name__ == '__main__':
    # parse arguments and create an instance
    instance = …
    main(instance)
```

7.2.4 Debuggen von Python Scripts in Visual Studio Code

Es wird davon ausgegangen, dass die Visual Studio Code (VS Code) Installation auf aktuellem Stand ist und die **ms-python.python**-Erweiterung installiert ist. Einen allgemeinen Überblick dazu finden Sie unter Python debug configurations in Visual Studio Code.

Beachten Sie die folgenden Punkte:

- In dieser Anleitung wird **raptorxml-python** als Befehl verwendet, um RaptorXML Server als Python Interpreter auszuführen.
- Sie finden die ausführbare **raptorxml-python**-Datei im Ordner **bin** Ihres RaptorXML Server-Applikationsordners.

Übersicht

Wir stellen hier zwei Methoden vor, um Python Scripts mittels VS Code in RaptorXML Server zu debuggen.

- Die Methode 1 funktioniert auch bei Servern und RaptorXML Python Callbacks (**--script**-Option).
- Bei der Methode 2 muss kein Quellcode geändert werden. Es handelt sich hierbei um einen modifizierten Aufruf von RaptorXML. Die Methode 2 funktioniert bei Servern und RaptorXML Python Callbacks (**--script**-Option) nicht.
- Beide Methoden funktionieren mit einem Standard-Pyhton Interpreter und dem importierten RaptorXML Python-Modul (**'import altova_api.v2 as altova'**).

Methode 1: Änderung des Quellcodes

Führen Sie die folgenden Schritte durch:

1. Starten Sie: **raptorxml-python -m pip install --upgrade debugpy**
2. Fügen Sie zu Ihrem Python-Quellcode die folgenden Zeilen hinzu:

   ```
   Python
   import debugpy
   debugpy.listen(5678)
   debugpy.wait_for_client()
   debugpy.breakpoint()
   ```

3. Kopieren Sie diese Launch-Konfiguration in VS Code **launch.json** (für die obigen Werte kann die Standardeinstellung verwendet werden) und wählen Sie sie für **Run** aus.

   ```
   json5
     {
       "name": "Python: Remote Attach",
       "type": "python",
       "request": "attach",
   ```



```
        "connect": {
        "host": "localhost",
        "port": 5678
        },
        "pathMappings": [
          {
            "localRoot": "${workspaceFolder}",
            "remoteRoot": "."
          }
        ]
      }
```

Sie können diese auch mit dem Menübefehl **Run->Add Configuration...->Python->Remote Attach** ausführen und dabei die Standardeinstellungen übernehmen.

4. Führen Sie Ihr Python Script (oder RaptorXML mit `--script` Callbacks) wie gewohnt aus.
5. Starten Sie das Debuggen (normalerweise mit dem Tastaturkürzel **F5**).

Methode 2: Verwendung einer modifizierten Befehlszeile

Führen Sie die folgenden Schritte durch:

1. Fügen Sie (wie in Methode 1 weiter oben) eine Launch-Konfiguration hinzu und wählen Sie diese für `Run` aus.
2. Setzen Sie in Ihrem Python Script einen Breakpoint.
3. Führen Sie den Befehl `raptorxml-python -m debugpy --listen 0.0.0.0:5678 --wait-for-client your-script-.py` aus.
4. Starten Sie das Debuggen (normalerweise mit dem Tastaturkürzel **F5**).

Anmerkung: Das Debuggen funktioniert normalerweise auch mit Containern und Remote Servern. Sie müssen in der Launch-Konfiguration den `host`-Schlüssel des `connect`-Eintrags ändern. Sie können auch andere Ports verwenden, solange der Code oder die Befehlszeile und `launch.json` einheitliche Werte haben.

Definieren von raptorxml-python.exe als Standard-Interpreter von VS Code

`raptorxml-python.exe` kann als Standard-Python Interpreter von VS Code konfiguriert werden. Fügen Sie dazu folgende Zeilen zu Ihrer VS Code-Datei `settigs.json` hinzu:

```
.json
  "python.defaultInterpreterPath": "/path/to/raptorxml-python.exe"
   ...
```

In diesem Fall kann auch eine Launch-Konfiguration "Current File" zum Starten des Scripts für das Debuggen verwendet werden. Nähere Informationen dazu finden Sie in der offiziellen Dokumentation zu VS Code.

7.2.5 FAQs

F: *Ich möchte ein Python Script schreiben, das bei Ausführung in Raptor Server für jedes Element eine eine neue XML-Instanz erstellt. Diese müssen je nach Parameter mit unterschiedlicher Kodierung und Formatierung in die Ausgabe serialisiert werden. Ist das in RaptorXML Server möglich?*

A: Nein, dies ist derzeit nicht möglich, da wir keine API zur Erstellung beliebiger XML-Instanzen haben. Für die Generierung von XBRL-Instanzen haben wir allerdings eine umfangreiche API, mit der zahlreiche technische Einzelheiten behandelt werden können (z.B. Vermeidung von doppelt vorhandenen contexts/units und vieles mehr). Nähere Informationen dazu finden Sie unter https://www.altova.com/manual/en/raptorapi/pyapiv2/2.11.0/html/xbrl.InstanceDocumentBuilder.html.

F: *Ich würde gerne lxml verwenden. Kann ich lxml-Bibliotheken unter "RaptorXMLXBRLServer2024/lib/" installieren?*

A: Sie können die meisten Python-Module mit dem folgenden Befehl direkt auf einem Terminal mit Administrator-Rechten installieren:

```
"/path/to/RaptorXML/application-folder/bin/RaptorXMLXBRL-python.exe" -m pip install lxml
```

F: *Wäre es in Ordnung, einen langen String zu erstellen, der die XML-Instanz enthält, das ganze Ding dann zu parsen und es erneut zu serialisieren?*

A: Das ist eine (von mehreren) Möglichkeiten. Sie können XML- und XBRL-Instanzen anhand eines String-Puffers folgendermaßen über die Python API parsen und validieren:

```
from altova_api.v2 import xml
txt = '''<?xml version="1.0" encoding="utf-8"?>
<doc>
    <elem attr="foo">bar</elem>
</doc>'''
inst = xml.Instance.create_from_buffer(txt.encode('utf-8')).result
print(inst.serialize())
```


7.3 .NET Framework API

Über die **.NET Framework API** von RaptorXML Server können Sie den **RaptorXML-Prozessor** in Applikationen integrieren, die in C# oder anderen .NET-Sprachen geschrieben wurden.

Sie ist als .NET Assembly implementiert und setzt den **RaptorXML-Prozessor** direkt in eine Applikation oder einen auf einem .NET-Framework basierenden Erweiterungsmechanismus wie VSTO (Visual Studio Tools for Office). Über die API haben sie umfassenden Zugriff auf Funktionen zum Validieren von Dokumenten und Abfragen des internen Datenmodells von RaptorXML Server aus.

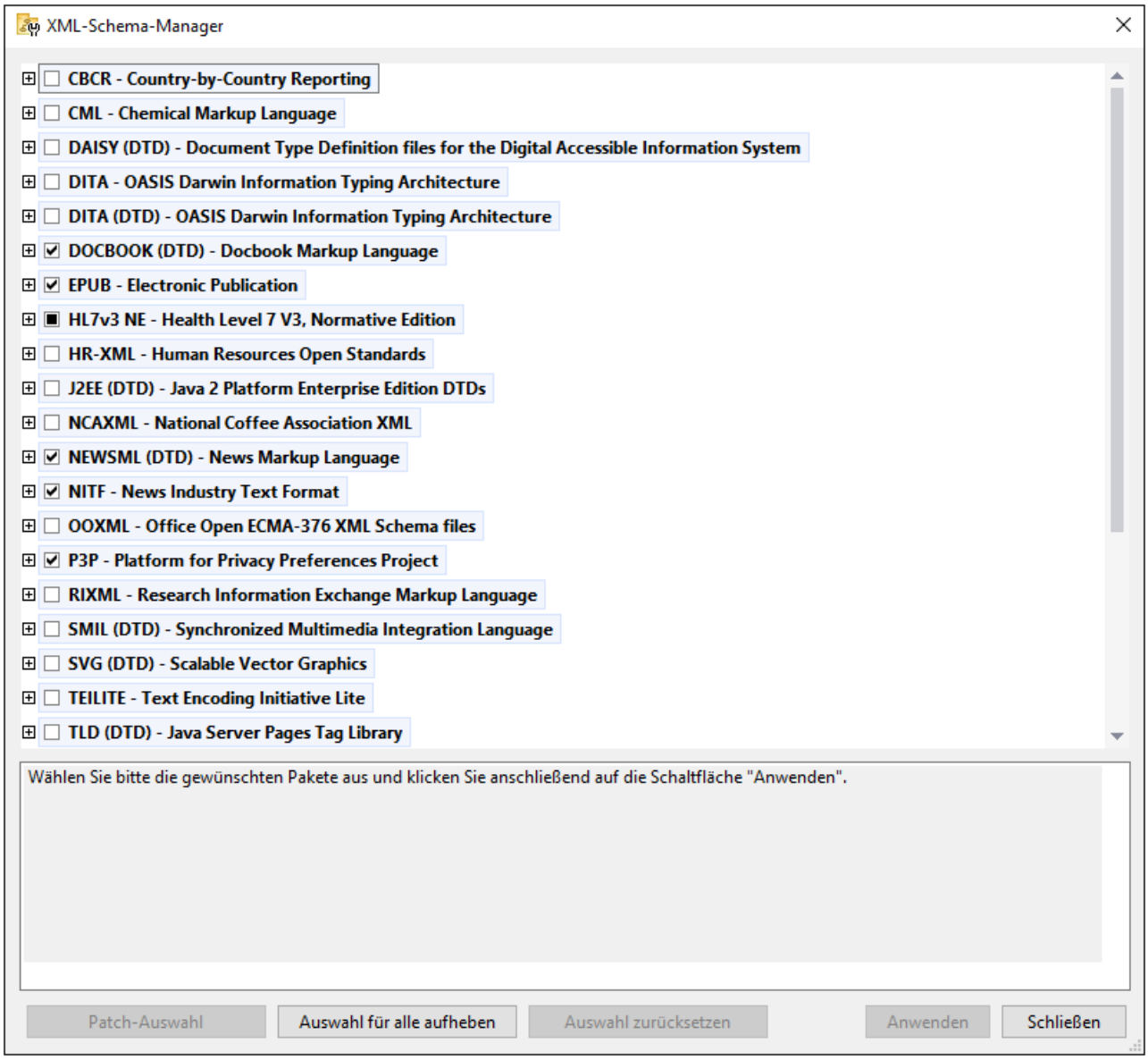
Verweise und Ressourcen

- *API-Dokumentation:* Die neueste RaptorXML Server .NET Framework API-Dokumentation finden Sie unter https://www.altova.com/manual/en/raptorapi/dotnetapiv2/2.11.0/html/index.html
- *Beispielcode:* Den Beispielcode finden Sie unter https://github.com/altova/RaptorXML-Examples.

8 Schema-Manager

Der XML-Schema-Manager ist ein Altova-Tool, mit dem Sie XML-Schemas (DTDs für XML-Dateien und XML-Schemas) zentral installieren und verwalten können, um diese in allen XML-Schema-fähigen Applikationen von Altova einschließlich RaptorXML Server verwenden zu können.

- Unter Windows hat der Schema-Manager eine grafische Benutzeroberfläche (*siehe Abbildung unten*) und steht auch über die Befehlszeile zur Verfügung. (Die Desktop-Applikationen von Altova stehen nur unter Windows zur Verfügung; *siehe Liste unten*).
- Unter Linux und MacOS steht der Schema-Manager nur über die Befehlszeile zur Verfügung. (Die Server-Applikationen von Altova stehen unter Windows, Linux und macOS zur Verfügung; *siehe Liste unten*).



Altova-Applikationen, die mit Schema-Manager arbeiten:

Desktop-Applikationen (nur Windows)	Server-Applikationen (Windows, Linux, macOS)
XMLSpy (alle Editionen)	RaptorXML Server, RaptorXML+XBRL Server
MapForce (alle Editionen)	StyleVision Server
StyleVision (alle Editionen)	
Authentic Desktop Enterprise Edition	

Installation und Deinstallation des Schema-Managers

Der Schema-Manager wird bei der ersten Installation einer neuen Version des Altova Mission Kit oder einer der XML-Schema-fähigen Applikationen von Altova (*siehe Tabelle oben*) automatisch installiert.

Ebenso wird er auch automatisch entfernt, wenn Sie die letzte XML-Schema-fähige Applikation von Altova auf Ihrem Rechner deinstallieren.

Schema-Manager-Funktionalitäten

Im Schema-Manager stehen die folgenden Funktionalitäten zur Verfügung:

- Anzeigen der auf Ihrem Rechner installierten XML-Schemas und Überprüfung, ob neue Versionen zum Download zur Verfügung stehen.
- Download neuer Versionen von XML-Schemas unabhängig vom Altova Produkt-Release-Zyklus. (Die Schemas werden von Altova online bereitgestellt und können über den Schema-Manager heruntergeladen werden).
- Installation oder Deinstallation jeder beliebigen (oder ggf. aller) der zahlreichen Versionen eines bestimmten Schemas.
- Ein XML-Schema kann Abhängigkeiten von anderen Schemas aufweisen. Bei der Installation oder Deinstallation eines bestimmten Schemas informiert Sie der Schema-Manager über davon abhängige Schemas und installiert bzw. entfernt diese ebenfalls automatisch.
- Der Schema-Manager ordnet Schema-Referenzen mit Hilfe des XML-Katalogs lokalen Dateien zu. Dadurch lassen sich große XML-Schemas schneller verarbeiten, als wenn sie sich unter einem entfernten Pfad befinden.
- Alle wichtigen Schemas werden über den Schema-Manager bereitgestellt und regelmäßig auf die jeweils neuesten Version aktualisiert. Dadurch können alle Ihre Schemas zentral verwaltet werden und stehen allen XML-Schema-fähigen Applikationen von Altova jederzeit zur Verfügung.
- Im Schema-Manager vorgenommene Änderungen werden für alle auf dem Rechner installierten Altova-Produkte wirksam.
- Wenn Sie versuchen ein Dokument in einem Altova-Produkt anhand eines nicht installierten aber über Schema-Manager verfügbaren Schemas zu validieren, wird das Schema automatisch installiert. Wenn das Schema-Paket jedoch Namespace-Zuordnungen enthält, wird das Schema nicht automatisch installiert; in diesem Fall müssen Sie Schema-Manager starten, das/die gewünschte(n) Paket(e) auswählen und die Installation starten. Wenn Ihre offene Altova-Applikation nach der Installation nicht automatisch neu gestartet wird, müssen Sie sie manuell neu starten.

Funktionsweise

Alle in Altova-Produkten verwendeten XML-Schemas werden von Altova online bereitgestellt. Dieser Speicher wird bei Veröffentlichung neuer Versionen der Schemas aktualisiert. Im Schema-Manager werden sowohl bei Aufruf über die Benutzeroberfläche als auch über das CLI Informationen über die neuesten verfügbaren Schemas angezeigt. Sie können die gewünschten Schemas dann über den Schema-Manager installieren, aktualisieren oder deinstallieren.

Schemas können vom Schema-Manager auch auf eine weitere Art installiert werden. Sie können ein Schemas und die davon abhängigen Schemas auf der Altova Website (https://www.altova.com/de/schema-manager) auswählen. Daraufhin wird auf der Website eine Datei des Typs **.altova_xmlschemas** mit Informationen über Ihre ausgewählten Schemas zum Download vorbereitet. Bei Doppelklick auf diese Datei oder bei Übergabe an den Schema-Manager über das CLI als Argument des Befehls `install` 415 installiert der Schema-Manager die ausgewählten Schemas.

Lokaler Cache: Überprüfung Ihrer Schemas

Alle Informationen über installierte Schemas werden in einem zentralen Cache-Verzeichnis auf Ihrem Rechner aufgezeichnet. Das Verzeichnis befindet sich hier:

Windows	`C:\ProgramData\Altova\pkgs\.cache`
Linux	`/var/opt/Altova/pkgs\.cache`
macOS	`/var/Altova/pkgs`

Dieses Cache-Verzeichnis wird regelmäßig mit dem neuesten Status der Schemas aus dem Online-Speicher von Altova aktualisiert. Diese Aktualisierungen finden unter den folgenden Bedingungen statt:

- bei jedem Start von Schema-Manager.
- Wenn Sie RaptorXML Server zum ersten Mal an einem bestimmten Kalendertag starten.
- Wenn RaptorXML Server länger als 24 Stunden geöffnet ist, findet alle 24 Stunden eine Aktualisierung des Cache statt.
- Sie können den Cache auch durch Ausführung des `update` 418-Befehls über die Befehlszeilenschnittstelle aktualisieren.

Der Schema-Manager kann somit Ihre installierten Schemas über den Cache ständig anhand der online verfügbaren Schemas auf der Altova Website überprüfen.

Nehmen Sie keine manuellen Änderungen am Cache vor!

Das lokale Cache-Verzeichnis wird automatisch auf Basis der installierten oder deinstallierten Schemas verwaltet; es sollte nicht manuell geändert oder gelöscht werden. Falls Sie den Schema-Manager je in seinen Originalzustand zurücksetzen möchten, (i) führen Sie den CLI-Befehl `reset` 416 der Befehlszeilenschnittstelle und (ii) anschließend den Befehl `initialize` 414 aus. (Führen Sie alternativ dazu den Befehl **reset** mit der Option **-i** aus).

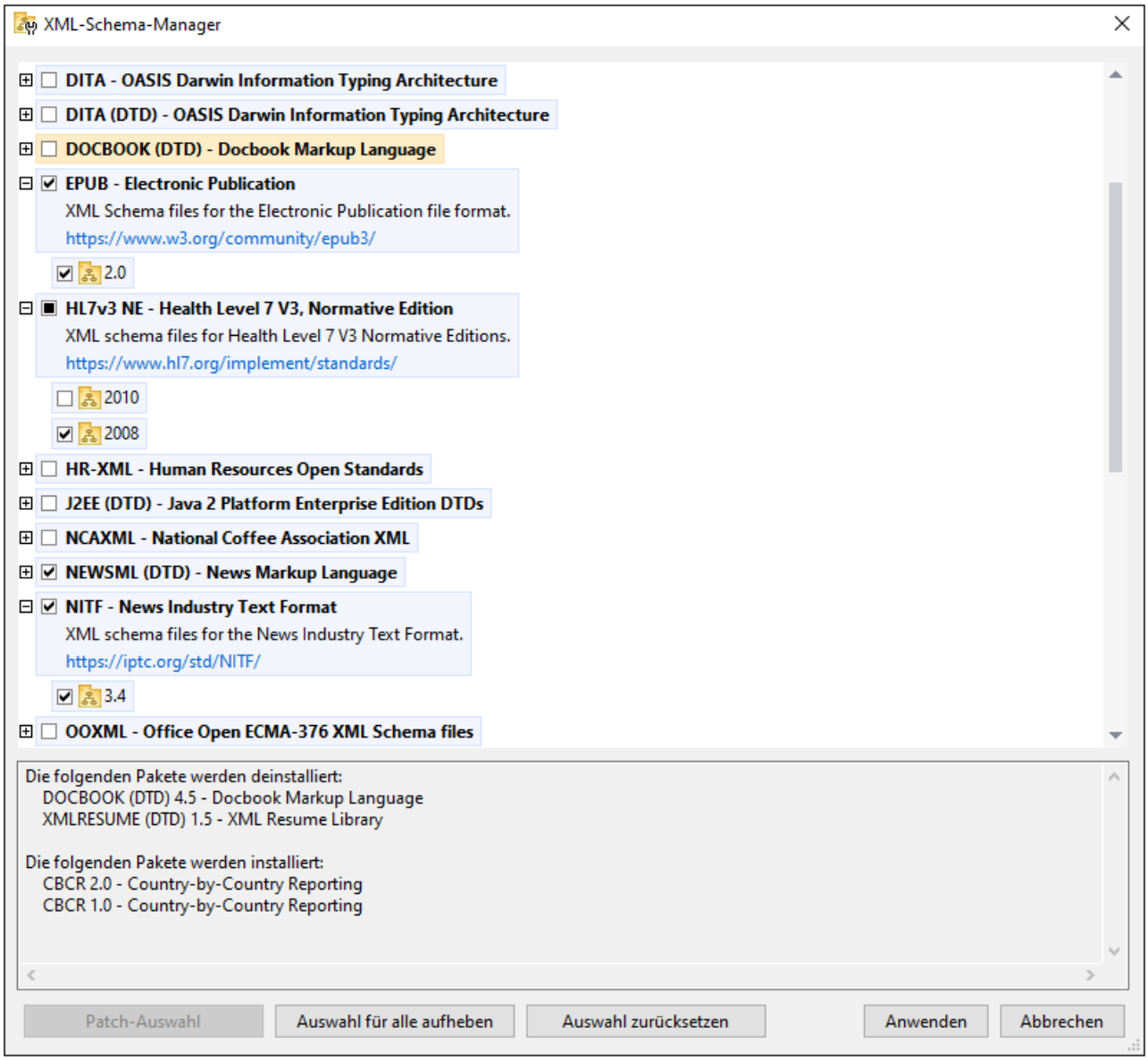
8.1 Ausführen des Schema-Managers

Grafische Benutzeroberfläche

Sie können die Benutzeroberfläche des Schema-Managers auf eine der folgenden Arten aufrufen:

- *Bei der Installationen von RaptorXML Server:* Aktivieren Sie gegen Ende der Installation das Kontrollkästchen *Altova-Schema-Manager aufrufen*, wodurch Sie die Benutzeroberfläche des Schema-Managers direkt aufrufen können. Auf diese Art können Sie Schemas während der Installation Ihrer Altova-Applikation installieren.
- über die von der Altova Webseite heruntergeladene `.altova_xmlschemas`-Datei: Doppelklicken Sie auf die heruntergeladene Datei, um den Schema-Manager zu starten, der daraufhin die (auf der Website) ausgewählten Schemas installiert.

Nachdem die Benutzeroberfläche des Schema-Managers geöffnet wurde (*Abbildung unten*), werden bereits installierte Schemas markiert angezeigt. Wenn ein zusätzliches Schema installiert werden soll, aktivieren Sie dieses. Wenn ein bereits installiertes Schema deinstalliert werden soll, deaktivieren Sie dieses. Nachdem Sie Ihre Auswahl getroffen haben, können Ihre Änderungen angewendet werden. Die Schemas, die installiert bzw. deinstalliert werden, werden markiert und im Fenster "Meldungen" am unteren Rand des Schema-Manager-Fensters (*siehe Abbildung*) erscheint eine Meldung über die bevorstehenden Änderungen.



Wenn Sie auf **Anwenden** klicken, wird der Fortschritt der Installation angezeigt. Bei Auftreten eines Fehlers (z.B. eines Verbindungsfehlers) wird eine Fehlermeldung angezeigt. Klicken Sie in diesem Fall auf die Schaltfläche **Erweitert**, die im Dialogfeld erscheint, überprüfen Sie die Schemaauswahl und versuchen Sie es erneut durch Klicken auf **Anwenden**.

Befehlszeilenschnittstelle

Sie können den Schema-Manager über eine Befehlszeilenschnittstelle starten, indem Sie Befehle an die ausführbare Datei `xmlschemamanager.exe` senden.

Die Datei `xmlschemamanager.exe` steht im folgenden Ordner zur Verfügung:

- *Unter Windows:* `C:\ProgramData\Altova\SharedBetweenVersions`

- *Unter Linux oder macOS (nur Server-Applikationen):* `%INSTALLDIR%/bin`, wobei `%INSTALLDIR%` das Installationsverzeichnis des Programms ist.

Anschließend können Sie jeden der im Abschnitt zur CLI-Befehlsreferenz 413 aufgelisteten Befehle verwenden.

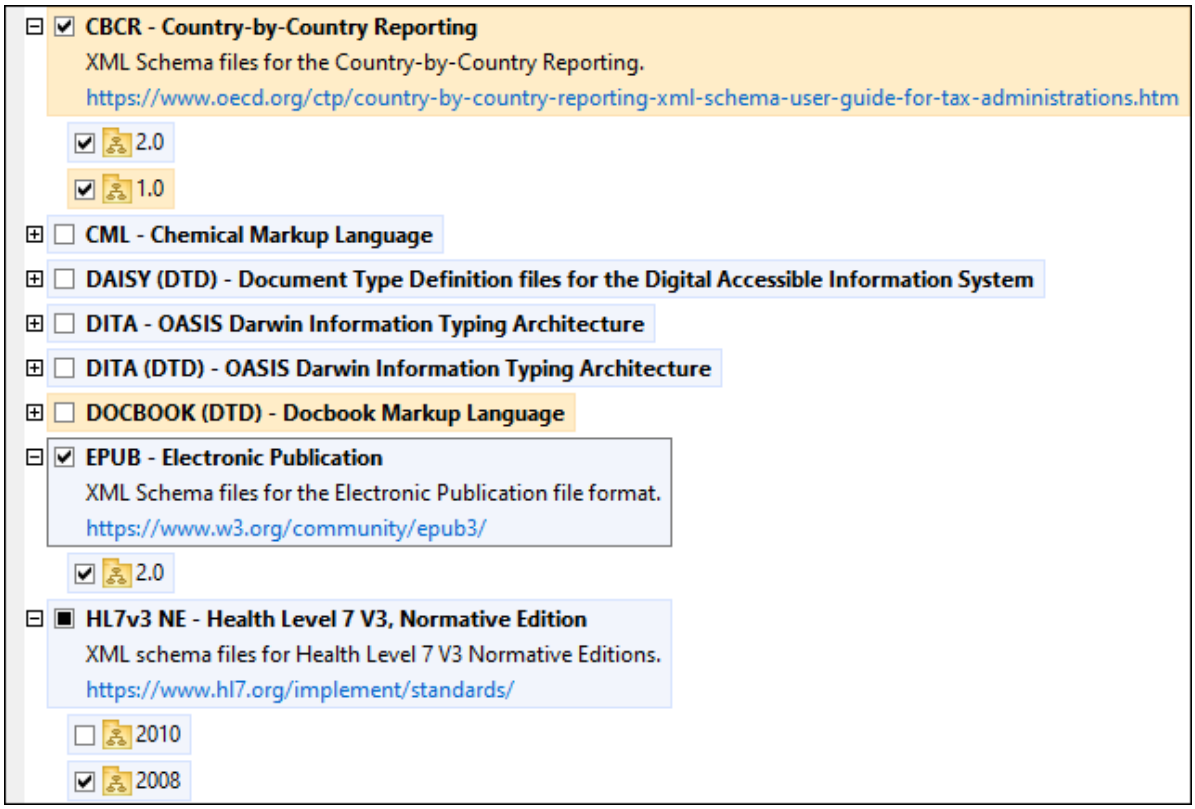
Um die Hilfe zu den Befehlen anzuzeigen, führen Sie den folgenden Befehl aus:

- *Unter Windows:* `xmlschemamanager.exe --help`
- *Unter Linux oder macOS (nur Server-Applikationen):* `sudo ./ xmlschemamanager --help`

8.2 Statuskategorien

Der Schema-Manager unterscheidet folgendermaßen zwischen den von ihm verwalteten Schemas:

- *installierte Schemas.* Diese werden auf der Benutzeroberfläche mit einem Häkchen angezeigt (*in der Abbildung unten sind die mit einem Häkchen versehenen und blau angezeigten Versionen der EPUB- und HL7v3 NE-Schemas installiert*). Wenn alle Versionen eines Schemas ausgewählt sind, wird ein Häkchen angezeigt. Wenn zumindest eine Version nicht ausgewählt ist, wird ein gefülltes Quadrat angezeigt. Sie können das Kontrollkästchen für ein installiertes Schema deaktivieren, um es zu **deinstallieren**; (*in der Abbildung unten ist die DocBook DTD installiert und wurde deaktiviert; sie wird daher für die Deinstallation vorbereitet*).
- *Nicht installierte verfügbare Schemas.* Diese werden auf der Benutzeroberfläche mit einem deaktivierten Kontrollkästchen angezeigt. Sie können die Schemas, die **installiert** werden sollen, auswählen.



- *Schemas, für die ein Upgrade zur Verfügung steht* sind diejenigen, die seit ihrer Installation vom Herausgeber überarbeitet wurden. Sie werden auf der Benutzeroberfläche durch ein -Symbol gekennzeichnet. Sie können für ein installiertes Schema ein **Patch** der verfügbaren überarbeiteten Version installieren.

Wichtige Punkte

- In der Abbildung oben sind beide CBCR-Schemas ausgewählt. Dasjenige mit einem blauen Hintergrund ist bereits installiert. Dasjenige mit dem gelben Hintergrund ist nicht installiert und wurde für die Installation ausgewählt. Beachten Sie, dass das Schema "HL7v3 NE 2010" nicht installiert ist und nicht für die Installation ausgewählt wurde.

- Ein gelber Hintergrund bedeutet, dass das Schema auf irgendeine Art geändert wird, wenn Sie auf die Schaltfläche **Anwenden** klicken. Wenn ein Schema deaktiviert ist und einen gelben Hintergrund hat, bedeutet dies, dass es bei Klick auf die Schaltfläche **Anwenden** deinstalliert wird. In der Abbildung oben ist dies bei der DocBook DTD der Fall.
- Bei Ausführung des Schema-Managers über die Befehlszeile wird der Befehl `list` 416 mit verschiedenen Optionen verwendet, um verschiedene Schemakategorien aufzulisten:

`xmlschemamanager.exe list`	Listet alle installierten und verfügbaren Schemas auf; auch verfügbare Upgrades werden angezeigt.
`xmlschemamanager.exe list -i`	Listet nur installierte Schemas auf; auch verfügbare Upgrades werden angezeigt.
`xmlschemamanager.exe list -u`	Listet Schemas auf, für die Upgrades zur Verfügung stehen

Anmerkung: Verwenden Sie unter Linux und macOS **`sudo ./xmlschemamanager list`**

8.3 Anwenden eines Patch oder Installation eines Schemas

Anwenden eines Patch auf ein installiertes Schema

Von Zeit zu Zeit werden von den Herausgebern der XML-Schemas Patches (Upgrades oder Überarbeitungen) veröffentlicht. Wenn der Schema-Manager erkennt, dass Patches zur Verfügung stehen, werden diese in der Schemaliste des Schema-Managers angezeigt und Sie können diese Patches schnell installieren.

Über die Benutzeroberfläche

Patches werden mit dem Symbol  gekennzeichnet. (*Siehe auch vorhergehendes Kapitel über Statuskategorien*[408]*).* Falls Patches zur Verfügung stehen, ist die Schaltfläche **Patch-Auswahl** aktiv. Klicken Sie darauf, um alle Patches für die Installation auszuwählen und vorzubereiten. Auf der Benutzeroberfläche ändert sich das Symbol von Schemas, für die ein Patch installiert wird von  in  , und im Fenster "Meldungen" am unteren Rand des Dialogfelds werden die Patches, die angewendet werden, aufgelistet. Sobald Sie mit der Auswahl fertig sind, klicken Sie auf **Anwenden**. Alle Patches werden gemeinsam angewendet. Beachten Sie, dass ein für die Installation eines Patch markiertes Schema deinstalliert wird, wenn Sie die Auswahl aufheben.

Über das CLI

So wenden Sie einen Patch über die Befehlszeilenschnittstelle an:

1. Führen Sie den Befehl `list -u`[416] aus. Daraufhin werden alle Schemas, für die Upgrades zur Verfügung stehen, aufgelistet.
2. Führen Sie den Befehl `upgrade`[419] aus, um alle Patches zu installieren.

Installieren eines verfügbaren Schemas

Sie können Schemas entweder über die Benutzeroberfläche des Schema-Managers oder durch Senden der Schema-Manager-Installationsbefehle über die Befehlszeile installieren.

Anmerkung: Wenn das aktuelle Schema andere Schemas referenziert, werden auch die referenzierten Schemas installiert.

Über die Benutzeroberfläche

Um Schemas über die Benutzeroberfläche des Schema-Managers zu installieren, wählen Sie die gewünschten Schemas aus und klicken Sie auf **Anwenden**.

Sie können die gewünschten Schemas auch auf der Altova Website auswählen und eine herunterladbare `.altova_xmlschemas`-Datei generieren. Bei Doppelklick auf diese Datei wird der Schema-Manager aufgerufen, in dem die gewünschten Schemas bereits vorausgewählt sind. Sie müssen nur mehr auf **Anwenden** klicken.

Über das CLI

Um Schemas über die Befehlszeile zu installieren, rufen Sie den Befehl `install`[415] auf:

```
xmlschemamanager.exe install [options] Schema+
```

wobei es sich bei `Schema` um das/die gewünschte(n) Schema(s) bzw. eine `.altova_xmlschemas`-Datei handelt. Ein Schema wird von einem Identifier im Format `<name>-<version>` referenziert. (Die Identifier von

Schemas werden angezeigt, wenn Sie den Befehl `list` [416] ausführen.) Sie können beliebig viele Schemas eingeben. Nähere Informationen dazu finden Sie unter der Beschreibung des Befehls `install` [415].

Anmerkung: Verwenden Sie unter Linux oder macOS den Befehl **`sudo ./xmlschemamanager`**.

Installation eines benötigten Schemas

Wenn Sie einen XML-Schema-Befehl in RaptorXML Server ausführen und RaptorXML Server erkennt, dass ein zur Ausführung des Befehls erforderliches Schema nicht vorhanden oder unvollständig ist, wird der Schema-Manager mit Informationen über das/die fehlende(n) Schema(s) aufgerufen. Sie können die gewünschten Schemas dann über den Schema-Manager direkt installieren.

Alle bereits installierten Schemas können jederzeit durch Aufruf des Schema-Managers über **Extras | Schema-Manager** über die Benutzeroberfläche des Schema-Managers angezeigt werden.

8.4 Deinstallieren eines Schemas, Zurücksetzen

Deinstallieren eines Schemas

Sie können Schemas entweder über die Benutzeroberfläche des Schema-Managers oder durch Senden der Schema-Manager-Deinstallationsbefehle über die Befehlszeile deinstallieren.

Anmerkung: Wenn das gewünschte Schema andere Schemas referenziert, so werden auch die referenzierten Schemas deinstalliert.

Über die Benutzeroberfläche

Um Schemas über die Benutzeroberfläche des Schema-Managers zu deinstallieren, deaktivieren Sie die Kontrollkästchen der entsprechenden Schemas und klicken Sie auf **Anwenden**. Daraufhin werden die ausgewählten Schemas und die davon referenzierten Schemas deinstalliert.

Um alle Schemas zu deinstallieren, klicken Sie auf **Auswahl für alle aufheben** und anschließend auf **Anwenden**.

Über das CLI

Um Schemas über die Befehlszeile zu deinstallieren, rufen Sie den Befehl `uninstall`[417] auf:

```
xmlschemamanager.exe uninstall [options] Schema+
```

wobei es sich beim Argument `Schema` ein zu deinstallierendes Schema oder eine `.altova_xmlschemas`-Datei handelt. Ein Schema wird von einem Identifier im Format `<name>-<version>` definiert. (Die Identifier von Schemas werden angezeigt, wenn Sie den Befehl `list`[416] ausführen.) Sie können beliebig viele Schemas eingeben. Nähere Informationen dazu finden Sie unter der Beschreibung des Befehls `uninstall`[417].

Anmerkung: Verwenden Sie unter Linux oder macOS den Befehl `sudo ./xmlschemamanager`.

Zurücksetzen des Schema-Managers

Sie können den Schema-Manager zurücksetzen. Damit werden alle installierten Schemas und das Cache-Verzeichnis entfernt.

- Klicken Sie auf der Benutzeroberfläche auf **Auswahl zurücksetzen**.
- Führen Sie über die Benutzeroberfläche den Befehl `reset`[416] aus.

Nachdem Sie diesen Befehl ausgeführt haben, muss der Befehl `initialize`[414] ausgeführt werden, um das Cache-Verzeichnis neu zu erstellen. Führen Sie alternativ dazu den Befehl `reset`[416] mit der Option `-i` aus.

Beachten Sie, dass mit `reset-i`[416] die Originalinstallation des Produkts wiederhergestellt wird, daher wird empfohlen, nach dem Zurücksetzen auch den Befehl `update`[418] auszuführen. Führen Sie alternativ dazu den Befehl `reset`[416] mit den Optionen `-i` und `-u` aus.

8.5 Befehlszeilenschnittstelle (CLI)

Um den Schema-Manager über die Befehlszeile aufzurufen, müssen Sie den Pfad zur ausführbaren Datei kennen. Standardmäßig befindet sich die ausführbare Schema-Manager-Datei hier:

Windows	`C:\ProgramData\Altova\SharedBetweenVersions\XMLSchemaManager.exe`
Linux	`/opt/Altova/RaptorXMLServer2025/bin/xmlschemamanager`
macOS	`/usr/local/Altova/RaptorXMLServer2025/bin/xmlschemamanager`

Anmerkung: Nachdem Sie auf Linux- und macOS-Systemen das Verzeichnis in dasjenige, das die ausführbare Datei enthält, geändert haben, können Sie die ausführbare Datei mit **`sudo ./xmlschemamanager`** aufrufen. Das Präfix **`./`** gibt an, dass sich die ausführbare Datei im aktuellen Verzeichnis befindet. Das Präfix **`sudo`** gibt an, dass der Befehl mit Root-Rechten ausgeführt werden muss.

Befehlszeilensyntax

Die allgemeine Syntax zur Verwendung der Befehlszeile lautet folgendermaßen:

```
<exec> -h | --help | --version | <command> [options] [arguments]
```

Der senkrechte Balken **`|`** im Codefragment oben trennt eine Gruppe einander gegenseitig ausschließender Elemente. Optionale Elemente stehen innerhalb von eckigen Klammern **`[]`**. Im Prinzip können Sie den Pfad zur ausführbaren Datei, gefolgt von entweder `--h`, `--help` oder `--version`-Optionen oder gefolgt von einem Befehl eingeben. Jeder Befehl kann Optionen und Argumente haben. Die Liste der Befehle wird in den folgenden Abschnitten beschrieben.

8.5.1 help

Mit diesem Befehl erhalten Sie Hilfe zu Befehlen zur ausführbaren Schema-Manager-Datei.

Syntax

```
<exec> help [Befehl]
```

`[Befehl]` ist hierbei ein optionales Argument zur Angabe jedes beliebigen gültigen Befehlsnamens.

Beachten Sie dazu Folgendes:

- Sie können die Hilfe zu einem Befehl auch durch Eingabe des Befehls, gefolgt von **`-h`** oder **`--help`** aufrufen, z.B: **`<exec> list -h`**
- Wenn Sie **`-h`** oder **`--help`** direkt nach dem Namen der ausführbaren Datei und vor einem Befehl eingeben, wird die allgemeine Hilfe (und nicht die Hilfe zu einem bestimmten Befehl) angezeigt, z.B: **`<exec> -h list`**

Beispiel

Mit dem folgenden Befehl wird Hilfe zum Befehl `list` angezeigt:

```
xmlschemamanager help list
```

8.5.2 info

Mit diesem Befehl werden ausführliche Informationen über die einzelnen als `Schema`-Argument angegebenen Schemas angezeigt. Darin enthalten sind Titel, Version, Beschreibung, Herausgeber der jeweils angegebenen Schemas und davon referenzierte Schemas sowie die Information, ob das Schema installiert ist oder nicht.

Syntax

```
<exec> info [options] Schema+
```

- Das Argument **`Schema`** ist der Name eines Schemas oder Teil eines Schemanamens. (Die Paket-ID eines Schemas und detaillierte Informationen über ihren Installationsstatus erhalten Sie mit dem Befehl `list` 416.)
- Mit **`<exec> info -h`** können Sie die Hilfe zum Befehl anzeigen.

Beispiel

Mit dem folgenden Befehl werden Informationen über das jeweils neueste **`DocBook-DTD`**- und **`NITF`**-Schemas angezeigt.

```
xmlschemamanager info doc nitf
```

8.5.3 initialize

Mit diesem Befehl wird die Schema-Manager-Umgebung initialisiert. Sie erstellen damit ein Cache-Verzeichnis, in dem Informationen über alle Schemas lokal gespeichert werden. Die Initialisierung erfolgt automatisch bei der ersten Installation einer Schema-fähigen Altova-Applikation. Normalerweise muss dieser Befehl nicht ausgeführt werden. Nach Ausführung des **`reset`**-Befehls ist dies allerdings erforderlich.

Syntax

```
<exec> initialize | init [options]
```

Optionen

Für den Befehl **`initialize`** stehen die folgenden Optionen zur Verfügung:

`--silent, --s`	Nur Fehlermeldungen anzeigen. Der Standardwert ist **`false`**.
`--verbose, --v`	Anzeige detaillierter Informationen während der Ausführung. Der Standardwert ist **`false`**.

`--help, --h`	Anzeige der Hilfe zum Befehl.

Beispiel

Mit dem folgenden Befehl wird der Schema-Manager initialisiert:

```
xmlschemamanager initialize
```

8.5.4 install

Mit diesem Befehl installieren Sie ein oder mehrere Schemas.

Syntax

```
<exec> install [options] Schema+
```

Um mehrere Schemas zu installieren, fügen Sie das Argument **`Schema`** mehrmals hinzu.

Als **`Schema`**-Argument kann eines der folgenden verwendet werden:

- Ein Schema-Identifier (im Format **`<name>-<version>`**, z.B: **`cbcr-2.0`**). Um die Schema-Identifier der gewünschten Schemas zu eruieren, führen Sie den Befehl `list`[416] aus. Sie können auch einen abgekürzten Identifier verwenden, sofern dieser eindeutig ist, z.B. **`docbook`**. Falls Sie einen abgekürzten Identifier verwenden, wird die neueste Version dieses Schemas installiert.
- Der Pfad zu einer von der Altova-Website heruntergeladenen **`.altova_xmlschemas`**-Datei. Informationen zu diesen Dateien finden Sie in der *Einführung zu Schema-Manager: Funktionsweise*[402].

Optionen

Für den Befehl **`install`** stehen die folgenden Optionen zur Verfügung:

`--silent, --s`	Nur Fehlermeldungen anzeigen. Der Standardwert ist **`false`**.
`--verbose, --v`	Anzeige detaillierter Informationen während der Ausführung. Der Standardwert ist **`false`**.
`--help, --h`	Anzeige der Hilfe zum Befehl.

Beispiel

Mit dem folgenden Befehl werden das CBCR 2.0 (Country-By-Country Reporting)-Schema und die neueste DocBook-DTD installiert:

```
xmlschemamanager install cbcr-2.0 docbook
```

8.5.5 list

Mit diesem Befehl werden vom Schema-Manager verwaltete Schemas aufgelistet. In der Liste wird eine der folgenden Informationen angezeigt:

- alle verfügbaren Schemas
- Schemas, die im Namen den im Argument **Schema** angegebenen String enthalten
- nur installierte Schemas
- Nur Schemas, für die ein Upgrade installiert werden kann

Syntax

```
<exec> list | ls [options] Schema?
```

Wenn kein **Schema**-Argument angegeben wird, werden alle verfügbaren Schemas aufgelistet. Andernfalls werden die durch die angegebenen Optionen definierten Schemas aufgelistet (*siehe Beispiel unten*). Beachten Sie, dass Sie das Argument **Schema** mehrfach angeben können.

Optionen
Für den Befehl **list** stehen die folgenden Optionen zur Verfügung:

`--installed, --i`	Auflisten nur der installierten Schemas. Der Standardwert ist **false**.
`--upgradeable, --u`	Auflisten nur derjenigen Schemas, für die Upgrades (Patches) zur Verfügung stehen. Der Standardwert ist **false**.
`--help, --h`	Anzeige der Hilfe zum Befehl.

Beispiele

- Um alle verfügbaren Schemas aufzulisten, führen Sie den folgenden Befehl aus: **xmlschemamanager list**
- Um nur installierte Schemas aufzulisten, führen Sie **xmlschemamanager list -i** aus.
- Um Schemas, die in ihrem Namen entweder "doc" oder "nitf" enthalten, aufzulisten, führen Sie **xmlschemamanager list doc nitf** aus.

8.5.6 reset

Mit diesem Befehl werden alle installierten Schemas und das Cache-Verzeichnis entfernt. Ihre Schemaumgebung wird vollständig zurückgesetzt. Nachdem Sie diesen Befehl ausgeführt haben, muss der Befehl initialize[414] ausgeführt werden, um das Cache-Verzeichnis neu zu erstellen. Führen Sie alternativ dazu den Befehl **reset** mit der Option **-i** aus. Da mit **reset-i** die Originalinstallation des Produkts wiederhergestellt wird, wird empfohlen, nach dem Zurücksetzen und Initialisieren auch den Befehl update[418] auszuführen. Führen Sie alternativ dazu den Befehl **reset** mit den Optionen **-i** und **-u** aus.

Syntax

```
<exec> reset [Optionen]
```

Optionen
Für den Befehl **reset** stehen die folgenden Optionen zur Verfügung:

`--init, --i`	Initialisierung des Schema-Managers nach dem Zurücksetzen. Der Standardwert ist **false**.
`--update, --u`	Aktualisiert die Liste der verfügbaren Schemas im Cache. Der Standardwert ist **false**.
`--silent, --s`	Nur Fehlermeldungen anzeigen. Der Standardwert ist **false**.
`--verbose, --v`	Anzeige detaillierter Informationen während der Ausführung. Der Standardwert ist **false**.
`--help, --h`	Anzeige der Hilfe zum Befehl.

Beispiele

- Um den Schema-Manager zurückzusetzen, führen Sie den folgenden Befehl aus: **xmlschemamanager reset**
- Um den Schema-Manager zurückzusetzen und ihn zu initialisieren, führen Sie **xmlschemamanager reset -i** aus.
- Um den Schema-Manager zurückzusetzen, ihn zu initialisieren und seine Schemaliste zu aktualisieren, führen Sie **xmlschemamanager reset -i -u** aus.

8.5.7 uninstall

Mit diesem Befehl deinstallieren Sie ein oder mehrere Schemas. Standardmäßig werden auch alle Schemas, die vom der aktuellen Schema referenziert werden, deinstalliert. Um nur das aktuelle Schema zu deinstallieren und die referenzierten Schemas beizubehalten, setzen Sie die Option `--k`.

Syntax

```
<exec> uninstall [options] Schema+
```

Um mehrere Schemas zu deinstallieren, fügen Sie das Argument **Schema** mehrmals hinzu.

Als **Schema**-Argument kann eines der folgenden verwendet werden:

- Ein Schema-Identifier (im Format **<name>-<version>**, z.B: **cbcr-2.0**). Um die Schema-Identifier der installierten Schemas zu eruieren, führen Sie den Befehl list -i 416 aus. Sie können auch einen abgekürzten Schemanamen verwenden, sofern dieser eindeutig ist, z.B. **docbook**. Falls Sie einen abgekürzten Namen verwenden, werden alle Schemas, die die Abkürzung in ihrem Namen enthalten, deinstalliert.

- Der Pfad zu einer von der Altova-Website heruntergeladenen **.altova_xmlschemas**-Datei. Informationen zu diesen Dateien finden Sie in der *Einführung zu Schema-Manager: Funktionsweise*[402].

Optionen
Für den Befehl **uninstall** stehen die folgenden Optionen zur Verfügung:

`--keep-references, --k`	Definieren Sie diese Option, um referenzierte Schemas beizubehalten. Der Standardwert ist **false**.
`--silent, --s`	Nur Fehlermeldungen anzeigen. Der Standardwert ist **false**.
`--verbose, --v`	Anzeige detaillierter Informationen während der Ausführung. Der Standardwert ist **false**.
`--help, --h`	Anzeige der Hilfe zum Befehl.

Beispiel

Mit dem folgenden Befehl werden die Schemas CBCR 2.0 und EPUB 2.0 und deren Abhängigkeiten deinstalliert:

```
xmlschemamanager uninstall cbcr-2.0 epub-2.0
```

Mit dem folgenden Befehl wird das **eba-2.10**-Schema, nicht aber die davon referenzierten Schemas deinstalliert:

```
xmlschemamanager uninstall --k cbcr-2.0
```

8.5.8 update

Mit diesem Befehl wird die Liste der über den Online-Speicher verfügbaren Schemas abgefragt und das lokale Cache-Verzeichnis wird aktualisiert. Normalerweise muss dieser Befehl nur ausgeführt werden, wenn Sie `reset`[416] und `initialize`[414] ausgeführt haben.

Syntax

```
<exec> update [options]
```

Optionen
Für den Befehl **update** stehen die folgenden Optionen zur Verfügung:

`--silent, --s`	Nur Fehlermeldungen anzeigen. Der Standardwert ist **false**.
`--verbose, --v`	Anzeige detaillierter Informationen während der Ausführung. Der Standardwert ist **false**.
`--help, --h`	Anzeige der Hilfe zum Befehl.

Beispiel

Mit dem folgenden Befehl wird der lokale Cache mit der Liste der neuesten Schemas aktualisiert:

```
xmlschemamanager update
```

8.5.9 upgrade

Mit diesem Befehl werden alle installierten Schemas, für die ein Upgrade installiert werden kann, auf die neueste verfügbare *Patch*-Version aktualisiert. Um herauszufinden, welche Schemas aktualisiert werden können, starten Sie den Befehl list-u 416.

Anmerkung: Der Befehl **upgrade** entfernt ein veraltetes Schema, falls keine neuere Version zur Verfügung steht.

Syntax

```
<exec> upgrade [Optionen]
```

Optionen
Für den Befehl **upgrade** stehen die folgenden Optionen zur Verfügung:

`--silent, --s`	Nur Fehlermeldungen anzeigen. Der Standardwert ist **false**.
`--verbose, --v`	Anzeige detaillierter Informationen während der Ausführung. Der Standardwert ist **false**.
`--help, --h`	Anzeige der Hilfe zum Befehl.

9 Zusätzliche Informationen

Dieser Abschnitt enthält die folgenden zusätzlichen Informationen:

9.1 Exitcodes

Es stehen die folgenden Exitcodes zur Verfügung:

0	Die Validierung war erfolgreich.
1	Validierung ist mit einem Fehler fehlgeschlagen / der Vorgang wurde mit Strg+C/Pause/Schließen des Terminals beendet / die Lizenz ist während der Ausführung abgelaufen.
11	RaptorXML konnte nicht gestartet werden; den Grund dafür finden Sie in der Log-Datei
22	Root-Katalog konnte nicht geladen werden / Listendatei konnte nicht geladen werden
64	Ungültiger Befehl/ungültige Optionen
77	Es konnte beim Start keine Lizenz abgerufen werden.
128+**n**	RaptorXML wurde aufgrund der Signalzahl **n** beendet. Alle Exit-Codes über **128** weise auf eine Beendung infolge eines extern empfangenen oder intern ausgelösten Signals hin. Wenn der Exitcode z.B. **134** lautet, so ist die Signalzahl **134-128=6** (die Zahl von **SIGABRT**).

9.2 Hinweise zum Schemapfad

In Instanzdokumenten können Hinweise zur Angabe des Schemapfads angegeben werden. Dafür stehen zwei Attribute zur Verfügung:

- `xsi:schemaLocation` für Schemadokumente mit Ziel-Namespaces. Beim Wert des Attributs handelt es sich um ein Elementpaar. Das erste ist ein Namespace, das zweite eine URL, unter dem sich ein Schemadokument befindet. Der Namespace-Name muss mit dem Ziel-Namespace des Schemadokuments übereinstimmen..

```
<document xmlns="http://www.altova.com/schemas/test03"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://www.altova.com/schemas/test03 Test.xsd">
```

- `xsi:noNamespaceSchemaLocation` für Schemadokumente ohne Ziel-Namespaces. Der Wert des Attributs ist die URL des Schemadokuments. Das referenzierte Schemadokument darf keinen Ziel-Namespace haben.

```
<document xmlns="http://www.altova.com/schemas/test03"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:noNamespaceSchemaLocation="Test.xsd">
```

Die Option `--schemalocation-hints` gibt an, wie diese beiden Attribute als Hinweise zu verwenden sind, v.a. wie die Informationen des `schemaLocation` Attributs zu behandeln sind (*siehe Optionsbeschreibung oben*). Beachten Sie, dass RaptorXML Server den Namespace-Teil des `xsi:noNamespaceSchemaLocation` Werts als leeren String behandelt.

Hinweise zum Schemapfad können auch in einer `import` Anweisung eines XML-Schema-Dokuments angegeben werden.

```
<import namespace="someNS" schemaLocation="someURL">
```

Auch in der `import` Anweisung können über einen Namespace, der auf ein Schema in einer Katalogdatei gemappt werden oder direkt als URL in einem `schemaLocation` Attribut angegeben werden kann, Hinweise gegeben werden. Die Option `--schema-imports` 255 gibt (für XBRL und XSD/XML) an, wie der Schemapfad auszuwählen ist.

10 Informationen zu den Prozessoren

Dieser Abschnitt enthält Informationen zu den in RaptorXML Server enthaltenen XSLT- und XQuery-Prozessoren. Diese Informationen beziehen sich großteils auf das Verhalten der Prozessoren in Fällen, in denen die Spezifikationen von der Entscheidung, die das Verhalten bis zur Implementierung betrifft, abweichen. Des Weiteren enthält dieser Abschnitt Informationen über die Altova-Erweiterungsfunktionen für XPath/XQuery.

10.1 Informationen zum XSLT- und XQuery-Prozessor

Der XSLT- und der XQuery-Prozessor von RaptorXML Server hält sich genau an die W3C-Spezifikationen und ist daher strenger als die früheren Altova-Prozessoren, wie z.B. die in frühere Versionen von XMLSpy integrierten und die in AltovaXML, das Vorgängerprodukt von RaptorXML, integrierten. Infolgedessen werden auch leichte Fehler, die von früheren Prozessoren ignoriert wurden, von RaptorXML Server als Fehler gekennzeichnet.

Zum Beispiel:

- Wenn das Ergebnis eines Pfad-Operators sowohl Nodes als auch Nicht-Nodes enthält, wird ein Typfehler (`err:XPTY0018`) ausgegeben.
- Wenn `E1` in einem Pfadausdruck `E1/E2` nicht zu einer Node-Sequenz ausgewertet wird, wird ein Typfehler (`err:XPTY0019`) ausgegeben.

Ändern Sie bei Auftreten eines solchen Fehlers je nach Bedarf, entweder das XSLT/XQuery-Dokument oder das Instanzdokument.

In diesem Abschnitt sind implementierungsspezifische Funktionalitäten der Prozessoren geordnet nach Spezifikation beschrieben:

10.1.1 XSLT 1.0

Der XSLT 1.0-Prozessor von RaptorXML Server entspricht der XSLT 1.0 Recommendation vom 16. November 1999 und der XPath 1.0 Recommendation vom 16. November 1999 des World Wide Web Consortium (W3C). Beachten Sie die folgenden Informationen zur Implementierung.

Anmerkungen zur Implementierung

Wenn das `method`-Attribut von `xsl:output` auf HTML gesetzt ist oder wenn standardmäßig die HTML-Ausgabe ausgewählt ist, werden Sonderzeichen in der XML- oder XSLT-Datei als HTML-Zeichenreferenzen in das HTML-Ausgabedokument eingefügt. So wird z.B. das Zeichen U+00A0 (die hexadezimale Zeichenreferenz für ein geschütztes Leerzeichen) entweder als Zeichenreferenz (` ` or ` `) oder als Entity-Referenz ` ` in den HTML-Code eingefügt.

10.1.2 XSLT 2.0

In diesem Abschnitt:

Standardkonformität

Der XSLT 2.0-Prozessor von RaptorXML Server entspricht der XSLT 2.0 Recommendation vom 23. Jänner 2007 und der XPath 2.0 Recommendation vom 14. Dezember 2010 des World Wide Web Consortium (W3C).

Rückwärtskompatibilität

Der XSLT 2.0-Prozessor ist rückwärtskompatibel. Normalerweise kommt die Rückwärtskompatibilität des XSLT 2.0.-Prozessors nur dann zum Einsatz, wenn Sie den XSLT 2.0-Prozessor (CLI-Parameter `--xslt=2` 259) zur Verarbeitung eines XSLT 1.0 Stylesheets oder einer XSLT 1.0-Anweisung verwenden. Beachten Sie, dass sich das Ergebnis des XSLT 1.0-Prozessors und des rückwärtskompatiblen XSLT 2.0.Prozessors unter Umständen unterscheiden kann.

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 die folgenden Punkte:

- Der XSLT 2.0-Prozessor verwendet als **Standard-Funktions-Namespace** den Namespace für XPath 2.0- und XQuery 1.0-Funktionen (siehe Tabelle oben). Sie können daher XPath 2.0- und XSLT 2.0-Funktionen in Ihrem Stylesheet ohne Präfix verwenden. Wenn Sie den Namespace für XPath 2.0-Funktionen in Ihrem Stylesheet mit einem Präfix deklarieren, können Sie zusätzlich dazu das in der Deklaration zugewiesene Präfix verwenden.
- Bei Verwendung von Typ-Konstruktoren und Typen aus dem XML Schema-Namespace, muss bei Aufruf des Typ-Konstruktors (z.B. `xs:date`) das in der jeweiligen Namespace-Deklaration verwendeten Präfix verwendet werden.
- 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 XSLT 2.0-Prozessor ist schemafähig. Sie können daher benutzerdefinierte Schematypen und die `xsl:validate`-Anweisung verwenden.

Implementierungsspezifisches Verhalten

Im Folgenden finden Sie eine Beschreibung, wie der XSLT 2.0-Prozessor implementierungsspezifische Aspekte von bestimmten XSLT 2.0-Funktionen behandelt.

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

`function-available`
Die Funktion überprüft, ob in-scope-Funktionen (XSLT, XPath und Erweiterungsfunktionen) verfügbar sind.

`unparsed-text`
Das Attribut `href` akzeptiert (i) relative Pfade für Dateien im Basis-URI-Ordner und (ii) absolute Pfade mit oder ohne das `file://`-Protokoll. Zusätzlich werden die folgenden (Altova-spezifischen) Kodierungen unterstützt: `binarytobase16` und `binarytobase64`. Beispiel: `xs:base64Binary(unparsed-text('chart.png', 'x-binarytobase64'))`.

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

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

10.1.3 XSLT 3.0

Der XSLT 3.0-Prozessor von RaptorXML Server entspricht der XSLT 3.0 Recommendation vom 8. Juni 2017 und der XPath 3.1 Recommendation vom 21. März 2017 des World Wide Web Consortium (W3C).

Der XSLT 3.0-Prozessor hat dieselben implementierungsspezifischen Eigenschaften wie der XSLT 2.0-Prozessor[424]. Zusätzlich dazu unterstützt er eine Reihe neuer XSLT 3.0-Funktionen: XPath- und XQuery 3.1-Funktionen und Operatoren und die XPath 3.1-Spezifikation.

Anmerkung: Die optionale Streaming-Funktion wird derzeit nicht unterstützt., d.h. das gesamte Dokument wird unabhängig vom Wert des Attributs `streamable` in den Arbeitsspeicher geladen. Falls genügend Arbeitsspeicher zur Verfügung steht, (i) wird das gesamte Dokument - ohne Streaming - verarbeitet (ii) werden "guaranteed-streamable" Konstrukte korrekt verarbeitet, so als würde bei der Ausführung Streaming verwendet und (iii) Streaming-Fehler würden nicht erkannt werden. In 64-Bit-Apps sollte die Nicht-Streaming-Verarbeitung kein Problem darstellen. Falls zu wenig Arbeitsspeicher zur Verfügung steht, können Sie mehr Speicherplatz zum System hinzufügen.

Namespaces

In Ihrem XSLT 3.0 Stylesheet sollten die folgenden Namespaces deklariert sein, damit Sie alle in XSLT 3.0 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/XQuery 3.1-Funktionen	`fn:`	`http://www.w3.org/2005/xpath-functions`
Mathematische Funktionen	`math:`	`http://www.w3.org/2005/xpath-functions/math`
Zuordnungsfunktionen	`map:`	`http://www.w3.org/2005/xpath-functions/map`
Array-Funktionen	`array:`	`http://www.w3.org/2005/xpath-functions/array`
XQuery-, XSLT- und XPath-Fehlercodes	`err:`	`http://www.w3.org/2005/xpath-functions/xqt-errors`
Serialisierungsfunktionen	`Ausgabe`	`http://www.w3.org/2010/xslt-xquery-serialization`

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

```
<xsl:stylesheet version="3.0"
   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 die folgenden Punkte:

- Der XSLT 3.0-Prozessor verwendet als **Standard-Funktions-Namespace** den Namespace für XPath- und XQuery-Funktionen und Operatoren 3.1 (siehe Tabelle oben). Sie können daher die Funktionen dieses Namespace in Ihrem Stylesheet ohne Präfix verwenden. Wenn Sie den Namespace für Funktionen in Ihrem Stylesheet mit einem Präfix deklarieren, können Sie zusätzlich dazu das in der Deklaration zugewiesene Präfix verwenden.
- Bei Verwendung von Typ-Konstruktoren und Typen aus dem XML Schema-Namespace, muss bei Aufruf des Typ-Konstruktors (z.B. `xs:date`) das in der jeweiligen Namespace-Deklaration verwendeten Präfix verwendet werden.
- Einige XPath/XQuery-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')`.

10.1.4 XQuery 1.0

In diesem Abschnitt:

Standardkonformität

Der XQuery 1.0-Prozessor von RaptorXML Server entspricht der XQuery 1.0 Recommendation vom 14. Dezember 2010 des W3C. Der Query-Standard stellt bei vielen Funktionen frei, wie viele diese zu implementieren sind. Im Folgenden finden Sie eine Liste, wie der Altova XQuery 1.0-Prozessor diese Funktionen implementiert.

Schema-Fähigkeit

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

Kodierung

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`
Vordefinierte Funktionen	`fn:`	`http://www.w3.org/2005/xpath-functions`
Lokale Funktionen	`local:`	`http://www.w3.org/2005/xquery-local-functions`

Beachten Sie 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 vordefinierte Funktionen (siehe `fn:`) der Standard-Funktions-Namespace in XQuery ist, muss beim Aufruf von vordefinierten 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 vordefinierte Funktion aufzurufen, ohne die Namespace im Abfrage-Prolog deklarieren zu müssen (z.B.: `fn:string("Hello")`).
- Sie können den Standard-Funktions-Namespace durch Deklarierung des `default function namespace`-Ausdrucks im Abfrageprolog ändern.
- Bei Verwendung von Typen aus dem XML Schema-Namespace kann das Präfix `xs:` verwendet werden, ohne dass Sie den Namespace explizit deklarieren müssen und dieses Präfix im Abfrageprolog daran binden müssen. (Beispiel: `xs:date` und `xs:yearMonthDuration`.) Wenn Sie ein anderes Präfix für den XML-Schema-Namespace verwenden möchten, muss dieses im Abfrageprolog explizit deklariert werden. (Beispiel: `declare namespace alt = "http://www.w3.org/2001/XMLSchema"; alt:date("2004-10-04")`.)
- Beachten Sie, dass die Datentypen `untypedAtomic`, `dayTimeDuration` und `yearMonthDuration` mit den Candidate Recommendations vom 23 January 2007 aus dem XPath Datentypen-Namespace in den XML-Schema Namespace verschoben wurden, d.h. `xs:yearMonthDuration`.

Wenn Namespaces für Funktionen, Typ-Konstruktoren, 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 Auswertung der Abfrage verwendet.

Statische und dynamische Typüberprüfung

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. Wenn in dieser Phase ein Fehler gefunden wird, wird eine Meldung ausgegeben und die Ausführung wird gestoppt.

Die dynamische Typ-Überprüfung wird zur 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.")
```

External functions

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, die Strings auf Basis ihrer Unicode-Codepunkte vergleicht. Andere unterstützte Collations sind die hier[433] aufgelisteten ICU-Collations. Um eine bestimmte Collation zu verwenden, geben Sie ihre in der Liste der unterstützten Collations[433] angeführte URI an. String-Vergleiche, wie die Funktionen `fn:max` und `fn:min` werden anhand der angegebenen Collation durchgeführt. Wenn die Collation-Option nicht definiert ist, wird die Standard-Unicode Codepoint Collation verwendet.

Präzision von numerischen Typen

- Der Datentyp `xs:integer` hat eine beliebige Präzision, d.h. 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.

Implementierungsspezifisches Verhalten

Im Folgenden wird beschrieben, wie der XQuery- und der XQuery Update 1.0-Prozessor implementierungsspezifische Aspekte bestimmter Funktionen behandeln.

`unparsed-text`
Das Attribut `href` akzeptiert (i) relative Pfade für Dateien im Basis-URI-Ordner und (ii) absolute Pfade mit oder ohne das `file://`-Protokoll. Zusätzlich werden die folgenden (Altova-spezifischen) Kodierungen unterstützt: `binarytobase16` und `binarytobase64`. Beispiel: `xs:base64Binary(unparsed-text('chart.png', 'x-binarytobase64'))`.

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

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

10.1.5 XQuery 3.1

Der XQuery 3.1-Prozessor von RaptorXML Server entspricht der XQuery 3.1 Recommendation vom 21. März 2017 des World Wide Web Consortium (W3C) und unterstützt XPath- und XQuery-Funktionen 3.1. Die XQuery 3.1-Spezifikation ist eine Übermenge der 3.0-Spezifikation. Daher unterstützt der XQuery 3.1-Prozessor XQuery 3.0-Funktionen..

Namespaces

In Ihrem XQuery 3.1 Stylesheet sollten die folgenden Namespaces deklariert sein, damit Sie die in XQuery 3.1 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/XQuery 3.1-Funktionen	`fn:`	`http://www.w3.org/2005/xpath-functions`
Mathematische Funktionen	`math:`	`http://www.w3.org/2005/xpath-functions/math`
Zuordnungsfunktionen	`map:`	`http://www.w3.org/2005/xpath-functions/map`
Array-Funktionen	`array:`	`http://www.w3.org/2005/xpath-functions/array`
XQuery-, XSLT- und XPath-Fehlercodes	`err:`	`http://www.w3.org/2005/xpath-functions/xqt-errors`
Serialisierungsfunktionen	`output`	`http://www.w3.org/2010/xslt-xquery-serialization`

Beachten Sie die folgenden Punkte:

- Der Altova XQuery 3.1-Prozessor ist so konfiguriert, dass die oben aufgelisteten Präfixe an die entsprechenden Namespaces gebunden sind.
- Da der oben angeführte Namespace für vordefinierte Funktionen (siehe `fn:`) der Standard-Funktions-Namespace in XQuery ist, muss beim Aufruf von vordefinierten 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 vordefinierte Funktion aufzurufen, ohne die Namespace im Abfrage-Prolog deklarieren zu müssen (z.B.: `fn:string("Hello")`).
- Sie können den Standard-Funktions-Namespace durch Deklarierung des `default function namespace`-Ausdrucks im Abfrageprolog ändern.
- Bei Verwendung von Typen aus dem XML Schema-Namespace kann das Präfix `xs:` verwendet werden, ohne dass Sie den Namespace explizit deklarieren müssen und dieses Präfix im Abfrageprolog daran binden müssen. (Beispiel: `xs:date` und `xs:yearMonthDuration`.) Wenn Sie ein anderes Präfix für den XML-Schema-Namespace verwenden möchten, muss dieses im Abfrageprolog explizit deklariert werden. (Beispiel: `declare namespace alt = "http://www.w3.org/2001/XMLSchema"; alt:date("2004-10-04")`.)

Wenn Namespaces für Funktionen, Typ-Konstruktoren, 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.

Implementierungsspezifisches Verhalten

Die implementierungsspezifischen Eigenschaften sind dieselben wie für XQuery 1.0[428].

Zusätzlich dazu kann die Altova-spezifische Kodierung `x-base64tobinary` verwendet werden, um ein Binärdokument wie z.B. ein Bild zu erzeugen.

10.2 XSLT- und XPath/XQuery-Funktionen

Dieser Abschnitt enthält eine Liste von Altova-Erweiterungsfunktionen und anderen Erweiterungsfunktionen, die in XPath und/oder XQuery-Ausdrücken verwendet werden können. Itova-Erweiterungsfunktionen können mit dem XSLT- und XQuery-Prozessor von Altova verwendet werden und bieten zusätzliche Funktionalitäten zu den in den W3C-Standards definierten Funktionsbibliotheken.

In diesem Abschnitt werden hauptsächlich XPath/XQuery-Erweiterungsfunktionen, die von Altova für zusätzliche Operationen sowie andere Erweiterungsfunktionen[529] erstellt wurden, beschrieben. Diese Erweiterungsfunktionen[434] können vom Altova-XSLT- und XQuery-Prozessor gemäß den in diesem Abschnitt beschriebenen Regeln verarbeitet werden. Informationen zu den regulären XPath/XQuery-Funktionen finden Sie in der Altova XPath/XQuery-Funktionsreferenz

Allgemeine Punkte

Beachten Sie bitte die folgenden allgemeinen Punkte:

- Funktionen aus den in den W3C-Spezifikationen definierten core-Funktionsbibliotheken können ohne Präfix aufgerufen werden, da der Altova-XSLT- und XQuery-Prozessor Funktionen, die kein Präfix haben, als Funktionen des in der XPath/XQuery Functions-Spezifikation Standard-Funktions-Namespace `http://www.w3.org/2005/xpath-functions` liest. Wenn dieser Namespace in einem XSLT- oder XQuery-Dokument explizit deklariert ist, kann das in der Namespace-Deklaration definierte Präfix optional auch in Funktionsnamen verwendet werden.
- Wenn bei einer Funktion eine Sequenz von einem Datenelement als Argument erwartet wird und eine Sequenz von mehr als einem Datenelement gesendet wird, wird 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.

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 `implicit-timezone()` überprüft werden.

Collations

Die Standard-Collation ist die Unicode Codepoint Collation, die Strings auf Basis ihrer Unicode-Codepunkte vergleicht. Der Prozessor verwendet den Unicode Collation-Algorithmus. Andere unterstützte Collations sind die hier aufgelisteten ICU-Collations. Um eine bestimmte Collation zu verwenden, geben Sie Ihre URI an (siehe Tabelle unten). String-Vergleiche, wie die Funktionen `fn:max` und `fn:min` werden anhand der angegebenen Collation durchgeführt. Wenn die Collation-Option nicht definiert ist, wird die Standard-Unicode Codepoint Collation verwendet.

Sprache	URIs
da: Dänisch	da_DK
de: Deutsch	de_AT, de_BE, de_CH, de_DE, de_LI, de_LU
en: Englisch	en_AS, en_AU, en_BB, en_BE, en_BM, en_BW, en_BZ, en_CA, en_GB, en_GU, en_HK, en_IE, en_IN, en_JM, en_MH, en_MP, en_MT, en_MU, en_NA, en_NZ, en_PH, en_PK, en_SG, en_TT, en_UM, en_US, en_VI, en_ZA, en_ZW
es: Spanisch	es_419, es_AR, es_BO, es_CL, es_CO, es_CR, es_DO, es_EC, es_ES, es_GQ, es_GT, es_HN, es_MX, es_NI, es_PA, es_PE, es_PR, es_PY, es_SV, es_US, es_UY, es_VE
fr: Französisch	fr_BE, fr_BF, fr_BI, fr_BJ, fr_BL, fr_CA, fr_CD, fr_CF, fr_CG, fr_CH, fr_CI, fr_CM, fr_DJ, fr_FR, fr_GA, fr_GN, fr_GP, fr_GQ, fr_KM, fr_LU, fr_MC, fr_MF, fr_MG, fr_ML, fr_MQ, fr_NE, fr_RE, fr_RW, fr_SN, fr_TD, fr_TG
it: Italienisch	it_CH, it_IT
ja: Japanisch	ja_JP
nb: Norwegisch (Bokmal)	nb_NO
nl: Holländisch	nl_AW, nl_BE, nl_NL
nn: Norwegisch (Nynorsk)	nn_NO
pt: Portugiesisch	pt_AO, pt_BR, pt_GW, pt_MZ, pt_PT, pt_ST
ru: Russisch	ru_MD, ru_RU, ru_UA
sv: Schwedisch	sv_FI, sv_SE

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 `in-scope-prefixes()`, `namespace-uri()` und `namespace-uri-for-prefix()`.

10.2.1 Altova-Erweiterungsfunktionen

Altova-Erweiterungsfunktionen können in XPath/XQuery-Ausdrücken verwendet werden. Dadurch stehen neben den Funktionen in der Standardbibliothek der XPath-, XQuery- und XSLT-Funktionen zusätzliche Funktionen zur Verfügung. Die Altova-Erweiterungsfunktionen befinden sich im **Altova-Erweiterungsfunktions-Namespace, `http://www.altova.com/xslt-extensions`** und sind in diesem Abschnitt mit dem Präfix **`altova:`**, das als an diesen Namespace gebunden angenommen wird, gekennzeichnet. Beachten Sie, dass manche Funktionen in zukünftigen Versionen Ihres Produkts eventuell nicht mehr unterstützt werden oder dass sich das Verhalten einzelner Funktionen ändern kann. Um zu sehen, welche Altova-Erweiterungsfunktionen unterstützt werden, lesen Sie bitte die Dokumentation zur jeweiligen Release.

Die in der "XPath/XQuery Functions"-Spezifikation des W3C definierten Funktionen können (i) in einem XSLT-Kontext in XPath-Ausdrücken und (ii) in einem XQuery-Dokument in XQuery-Ausdrücken verwendet werden. In dieser Dokumentation sind die Funktionen, die im Zusammenhang mit XPath in XSLT verwendet werden können, mit einem **XP**-Symbol und Funktionen, die im Zusammenhang mit XQuery verwendet werden können, mit einem **XQ**-Symbol markiert; sie fungieren als XQuery-Funktionen. In den XSLT-Spezifikationen des W3C (nicht in den "XPath/XQuery Functions"-Spezifikationen) sind außerdem Funktionen definiert, die in XSLT-Dokumenten in XPath-Ausdrücken verwendet werden können. Diese Funktionen sind mit dem Symbol **XSLT** gekennzeichnet und werden als XSLT-Funktionen bezeichnet. In welcher XPath/XQuery- und XSLT-Version eine Funktion verwendet werden kann, wird in der Beschreibung der Funktion (*siehe Symbole unten)* angegeben. Funktionen aus der XPath/XQuery- und XSLT-Funktionsbibliothek werden ohne Präfix aufgelistet. Erweiterungsfunktionen aus anderen Bibliotheken wie z.B. Altova-Erweiterungsfunktionen werden mit einem Präfix angegeben.

XPath-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	**XP1** **XP2** **XP3.1**
XSLT-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	**XSLT1** **XSLT2** **XSLT3**
XQuery-Funktionen (in XQuery-Ausdrücken in XQuery verwendet):	**XQ1** **XQ3.1**

Verwendung von Altova-Erweiterungsfunktionen

Um Altova-Erweiterungsfunktionen verwenden zu können, müssen Sie den Altova-Erweiterungsfunktions-Namespace deklarieren (*erster markierter Bereich im Codefragment unten*) und die Erweiterungsfunktionen anschließend so verwenden, dass sie als zu diesem Namespace gehörig aufgelöst werden (*siehe zweiter markierter Bereich*). Im Beispiel unten wird die Altova-Erweiterungsfunktion **age** verwendet.

```
<xsl:stylesheet version="2.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:fn="http://www.w3.org/2005/xpath-functions"
    xmlns:altova="http://www.altova.com/xslt-extensions">
    <xsl:output method="text" encoding="ISO-8859-1"/>
    <xsl:template match="Persons">
        <xsl:for-each select="Person">
            <xsl:value-of select="concat(Name, ': ')"/>
            <xsl:value-of select="altova:age(xs:date(BirthDate))"/>
            <xsl:value-of select="' years&#x0A;'"/>
        </xsl:for-each>
    </xsl:template>
</xsl:stylesheet>
```

XSLT-Funktionen 436

XSLT-Funktionen können in XPath-Ausdrücken nur im XSLT-Kontext verwendet werden (ähnlich wie die XSLT 2.0-Funktionen `current-group()` oder `key()`). Diese Funktionen sind nicht für Nicht-XSLT-Kontext gedacht und funktionieren in einem solchen Kontext (z.B. in einem XQuery-Kontext) nicht. Beachten Sie, dass XSLT-Funktionen für XBRL nur mit Altova Produkten verwendet werden können, die XBRL unterstützen.

XPath/XQuery-Funktionen

XPath/XQuery-Funktionen können sowohl in XPath-Ausdrücken im XSLT-Kontext als auch in XQuery-Ausdrücken verwendet werden.

- Datum/Uhrzeit 439
- Standort 457
- Bildbezogene 469
- Numerisch 474
- Schema 476
- Sequenz 496
- String 505
- Verschiedenes 511

Diagrammfunktionen (nur Enterprise- und Server-Editionen)

Altova-Erweiterungsfunktionen für Diagramme 514 werden nur in der Enterprise und Server Edition von Altova-Produkten unterstützt und ermöglichen die Generierung von Diagrammen anhand von XML-Daten.

Barcode-Funktionen

Mit Hilfe der Altova Barcode-Erweiterungsfunktionen 526 können Barcodes generiert und in mittels XSLT Stylesheets generierte Ausgabedokumente eingefügt werden.

10.2.1.1 XSLT-Funktionen

XSLT-Erweiterungsfunktionen können in XPath-Ausdrücken in einem XSLT-Kontext verwendet werden. In einem Nicht-XSLT-Kontext (z.B. in einem XQuery-Kontext) funktionieren sie nicht.

Anmerkung zur Benennung von Funktionen und zur Anwendbarkeit der Sprache

Altova-Erweiterungsfunktionen können in XPath/XQuery-Ausdrücken verwendet werden. Dadurch stehen neben den Funktionen in der Standardbibliothek der XPath-, XQuery- und XSLT-Funktionen zusätzliche Funktionen zur Verfügung. Die Altova-Erweiterungsfunktionen befinden sich im **Altova-Erweiterungsfunktions-Namespace,** `http://www.altova.com/xslt-extensions` und sind in diesem Abschnitt mit dem Präfix `altova:`, das als an diesen Namespace gebunden angenommen wird, gekennzeichnet. Beachten Sie, dass manche Funktionen in zukünftigen Versionen Ihres Produkts eventuell nicht mehr unterstützt werden oder dass sich das Verhalten einzelner Funktionen ändern kann. Um zu sehen, welche Altova-Erweiterungsfunktionen unterstützt werden, lesen Sie bitte die Dokumentation zur jeweiligen Release.

XPath-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	XP1 XP2 XP3.1
XSLT-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	XSLT1 XSLT2 XSLT3
XQuery-Funktionen (in XQuery-Ausdrücken in XQuery verwendet):	XQ1 XQ3.1

Allgemeine Funktionen

▼ distinct-nodes [altova:]

altova:distinct-nodes(*node()**)* **als node()*** **XSLT1** **XSLT2** **XSLT3**
Erhält eine Gruppe von einem oder mehreren Nodes als Input und gibt dieselbe Gruppe ohne Nodes mit doppelt vorhandenen Werten zurück. Der Vergleich wird mittels der XPath/XQuery-Funktion `fn:deep-equal` durchgeführt.

⊟ *Beispiele*

- **altova:distinct-nodes**(country) gibt alle Child `country` Nodes ohne diejenigen mit doppelt vorhandenen Werten zurück.

▼ evaluate [altova:]

altova:evaluate(XPathExpression *as xs:string***[, ValueOf$p1, ... ValueOf$pN])** **XSLT1** **XSLT2** **XSLT3**
Erhält einen XPath-Ausdruck als obligatorisches Argument, der als String übergeben wird, und gibt das Resultat des ausgewerteten Ausdrucks zurück. Beispiel: **altova:evaluate('//Name[1]')** gibt den Inhalt des ersten `Name` Elements im Dokument zurück. Beachten Sie, dass der Ausdruck `//Name[1]` durch Einschließen in einfache Anführungszeichen als String übergeben wird.

Die Funktion `altova:evaluate` kann zusätzliche (optionale) Argumente erhalten. Diese Argumente sind die Werte der einzelnen im Geltungsbereich befindlichen Variablen und haben die Namen `p1`, `p2`, `p3`... `pN`. Beachten Sie zur Verwendung die folgenden Punkte: (i) Die Variablennamen müssen die Form `pX` haben, wobei `X` eine Ganzzahl ist; (ii) die Argumente der Funktion `altova:evaluate` (*siehe Signatur oben*) liefern vom zweiten Argument an die Werte der Variablen, wobei die Reihenfolge der Argumente der numerisch geordneten Variablensequenz entspricht: `p1` bis `pN`. Das zweite Argument wird der Wert der Variablen `p1`, das dritte Argument der der Variablen `p2`, usw.; (iii) Die Werte der Variablen müssen vom Typ `item*` sein

⊞ *Beispiel*

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

Beachten Sie im obigen Beispiel folgende Punkte:

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

⊟ *Weitere Beispiele für die Verwendung der Variablen*

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

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

Die `altova:evaluate()` Erweiterungsfunktion ist in Situationen nützlich, in denen ein XPath-Ausdruck im XSLT-Stylesheet einen oder mehrere Teile enthält, die dynamisch ausgewertet werden müssen. Angenommen ein Benutzer gibt seinen Request für das Sortierkriterium ein und das Sortierkriterium ist im Attribut `UserReq/@sortkey` gespeichert. Im Stylesheet könnten Sie den folgenden Ausdruck haben: `<xsl:sort select="altova:evaluate(../UserReq/@sortkey)" order="ascending"/>` Die `altova:evaluate()` Funktion liest das `sortkey` Attribut des `UserReq` Child-Elements des Parent des Kontext-Node. Angenommen der Wert des `sortkey` Attributs ist `Price`, dann wird von der `altova:evaluate()` Funktion `Price` zurückgegeben und wird zum Wert des `select` Attributs: `<xsl:sort select="Price" order="ascending"/>`. Wenn diese `sort` Anweisung im Kontext eines Elements namens `Order` vorkommt, dann werden die `Order` Elemente nach den Werten Ihrer `Price` Children sortiert. Alternativ dazu, wenn der Wert von `@sortkey` z.B. `Date` ist, werden die `Order` Elemente nach den Werten ihrer `Date` Children sortiert. Das Sortierkriterium für `Order` wird also zur Laufzeit aus dem `sortkey` Attribut ausgewählt. Diese hätte man mit einem Ausdruck wie dem folgenden nicht bewerkstelligen können: `<xsl:sort select="../UserReq/@sortkey" order="ascending"/>`. Im oben gezeigten Beispiel wäre das Sortierkriterium das `sortkey` Attribut selbst, nicht `Price` oder `Date` (oder jeder beliebige andere Inhalt von `sortkey`)

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

Weitere Beispiele

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

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

- Dynamischer XPath-Ausdruck ohne dynamische Variable:
  `<xsl:variable name="xpath" select="'$p3, $p2, $p1'" />`
  `<xsl:value-of select="altova:evaluate($xpath)" />`
  Gibt einen Fehler aus.: Es wurde keine Variable für `$p3` definiert.

encode-for-rtf [altova:]

`altova:encode-for-rtf(input` *`als xs:string`*`,` **`preserveallwhitespace`** *`als xs:boolean`*`,`
`preservenewlines` *`als xs:boolean`*`)` **`als xs:string`** **XSLT2** **XSLT3**
Konvertiert den Input-String in Code für RTF. Whitespaces und neue Zeilen werden gemäß dem für die entsprechenden Parameter definierten Booleschen Wert beibehalten.

[Nach oben 436]

XBRL-Funktionen

Altova XBRL-Funktionen können nur mit Editionen von Altova-Produkten verwendet werden, die XBRL unterstützen.

- xbrl-footnotes [altova:]

  altova:xbrl-footnotes(*node()***)** als **node()*** XSLT2 XSLT3
  Erhält einen Node als Input-Argument und gibt die durch den Input-Node referenzierte Gruppe der XBRL-Fußnoten-Nodes zurück.

- xbrl-labels [altova:]

  altova:xbrl-labels(*xs:QName***, *xs:string***)** als **node()*** XSLT2 XSLT3
  Erhält zwei Input-Argumente: einen Node-Namen und den Pfad der Taxonomiedatei, die den Node enthält. Die Funktion gibt die XBRL Labels zurück, die mit dem Input-Node verknüpft sind.

[Nach oben 436]

10.2.1.2 XPath/XQuery-Funktionen: Datum und Uhrzeit

Die Datums- und Uhrzeit-Erweiterungsfunktionen von Altova können im Zusammenhang mit XPath- und XQuery-Ausdrücken verwendet werden und bieten zusätzliche Funktionalitäten für die Verarbeitung von Daten, die in Form von XML-Schema-Datums- und Uhrzeit-Datentypen zur Verfügung stehen. Die Funktionen in diesem Abschnitt können mit dem **XPath 3.0** - und dem **XQuery 3.0** -Prozessor von Altova verwendet werden. Sie stehen in verschiedenen XPath/XQuery-Kontexten zur Verfügung.

Anmerkung zur Benennung von Funktionen und zur Anwendbarkeit der Sprache

Altova-Erweiterungsfunktionen können in XPath/XQuery-Ausdrücken verwendet werden. Dadurch stehen neben den Funktionen in der Standardbibliothek der XPath-, XQuery- und XSLT-Funktionen zusätzliche Funktionen zur Verfügung. Die Altova-Erweiterungsfunktionen befinden sich im **Altova-Erweiterungsfunktions-Namespace, `http://www.altova.com/xslt-extensions`** und sind in diesem Abschnitt mit dem Präfix **`altova:`**, das als an diesen Namespace gebunden angenommen wird, gekennzeichnet. Beachten Sie, dass manche Funktionen in zukünftigen Versionen Ihres Produkts eventuell nicht mehr unterstützt werden oder dass sich das Verhalten einzelner Funktionen ändern kann. Um zu sehen, welche Altova-Erweiterungsfunktionen unterstützt werden, lesen Sie bitte die Dokumentation zur jeweiligen Release.

XPath-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	XP1 XP2 XP3.1
XSLT-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	XSLT1 XSLT2 XSLT3
XQuery-Funktionen (in XQuery-Ausdrücken in XQuery verwendet):	XQ1 XQ3.1

- Nach Funktionalität gruppiert

▼ in alphabetischer Reihenfolge

[Nach oben 439]

Hinzufügen einer Zeitdauer zu xs:dateTime XP3.1 XQ3.1

Mit diesen Funktionen werden Zeitdauerwerte zu **xs:dateTime** hinzugefügt, bevor **xs:dateTime** zurückgegeben wird. Der Typ `xs:dateTime` hat das Format `JJJJ-MM-TTZhh:mm:ss.sss`. Es handelt sich hierbei um eine Verkettung des `xs:date` und `xs:time` Formats, getrennt durch den Buchstaben `Z`. Ein Zeitzonensuffix (wie z.B. `+01:00`) ist optional.

add-years-to-dateTime [altova:]

altova:add-years-to-dateTime(DateTime *als xs:dateTime*, **Years** *als xs:integer*) **als xs:dateTime** XP3.1 XQ3.1

Fügt eine Zeitdauer in Jahren zu einem `xs:dateTime` Wert (*siehe Beispiele unten*) hinzu. Beim zweiten Argument handelt es sich um die Anzahl der Jahre, die zu dem im ersten Parameter angegebenen `xs:dateTime` Wert hinzugefügt werden sollen. Das Ergebnis ist vom Typ `xs:dateTime`.

Beispiele

- **altova:add-years-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), 10) gibt 2024-01-15T14:00:00 zurück
- **altova:add-years-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), -4) gibt 2010-01-15T14:00:00 zurück

add-months-to-dateTime [altova:]

altova:add-months-to-dateTime(DateTime *als xs:dateTime*, **Months** *als xs:integer*) als **xs:dateTime** XP3.1 XQ3.1

Fügt eine Zeitdauer in Monaten zu einem `xs:dateTime` Wert (*siehe Beispiele unten*) hinzu. Beim zweiten Argument handelt es sich um die Anzahl der Monate, die zu dem im ersten Argument angegebenen `xs:dateTime` Wert hinzugefügt werden sollen. Das Ergebnis ist vom Typ `xs:dateTime`.

Beispiele

- **altova:add-months-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), 10) gibt 2014-11-15T14:00:00 zurück
- **altova:add-months-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), -2) gibt 2013-11-15T14:00:00 zurück

add-days-to-dateTime [altova:]

altova:add-days-to-dateTime(DateTime *als xs:dateTime*, **Days** *als xs:integer*) als **xs:dateTime** XP3.1 XQ3.1

Fügt eine Zeitdauer in Tagen zu einem `xs:dateTime` Wert (*siehe Beispiel unten*) hinzu. Beim zweiten Argument handelt es sich um die Anzahl der Tage, die zu dem im ersten Argument angegebenen `xs:dateTime` Wert hinzugefügt werden sollen. Das Ergebnis ist vom Typ `xs:dateTime`.

Beispiele

- **altova:add-days-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), 10) gibt 2014-01-25T14:00:00 zurück
- **altova:add-days-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), -8) gibt 2014-01-07T14:00:00 zurück

add-hours-to-dateTime [altova:]

altova:add-hours-to-dateTime(DateTime *als xs:dateTime*, **Hours** *als xs:integer*) als **xs:dateTime** **XP3.1** **XQ3.1**

Fügt eine Zeitdauer in Stunden zu einem `xs:dateTime` Wert (*siehe Beispiel unten*) hinzu. Beim zweiten Argument handelt es sich um die Anzahl der Stunden, die zu dem im ersten Argument angegebenen `xs:dateTime` Wert hinzugefügt werden sollen. Das Ergebnis ist vom Typ `xs:dateTime`.

Beispiele

- **altova:add-hours-to-dateTime**(xs:dateTime("2014-01-15T13:00:00"), 10) gibt 2014-01-15T23:00:00 zurück
- **altova:add-hours-to-dateTime**(xs:dateTime("2014-01-15T13:00:00"), -8) gibt 2014-01-15T05:00:00 zurück

add-minutes-to-dateTime [altova:]

altova:add-minutes-to-dateTime(DateTime *als xs:dateTime*, **Minutes** *als xs:integer*) als **xs:dateTime** **XP3.1** **XQ3.1**

Fügt eine Zeitdauer in Minuten zu einem `xs:dateTime` Wert (*siehe Beispiele unten)* hinzu. Beim zweiten Argument handelt es sich um die Anzahl der Minuten, die zu dem im ersten Argument angegebenen `xs:dateTime` Wert hinzugefügt werden sollen. Das Ergebnis ist vom Typ `xs:dateTime`.

Beispiele

- **altova:add-minutes-to-dateTime**(xs:dateTime("2014-01-15T14:10:00"), 45) gibt 2014-01-15T14:55:00 zurück
- **altova:add-minutes-to-dateTime**(xs:dateTime("2014-01-15T14:10:00"), -5) gibt 2014-01-15T14:05:00 zurück

add-seconds-to-dateTime [altova:]

altova:add-seconds-to-dateTime(DateTime *als xs:dateTime*, **Seconds** *als xs:integer*) als **xs:dateTime** **XP3.1** **XQ3.1**

Fügt eine Zeitdauer in Sekunden zu einem `xs:dateTime` Wert (*siehe Beispiele unten*) hinzu. Beim zweiten Argument handelt es sich um die Anzahl der Sekunden, die zu dem im ersten Argument angegebenen `xs:dateTime` Wert hinzugefügt werden sollen. Das Ergebnis ist vom Typ `xs:dateTime`.

Beispiele

- **altova:add-seconds-to-dateTime**(xs:dateTime("2014-01-15T14:00:10"), 20) gibt 2014-01-15T14:00:30 zurück
- **altova:add-seconds-to-dateTime**(xs:dateTime("2014-01-15T14:00:10"), -5) gibt 2014-01-15T14:00:05 zurück

[Nach oben 439]

Hinzufügen einer Zeitdauer zu xs:date XP3.1 XQ3.1

Mit diesen Funktionen werden Zeitdauerwerte zu **xs:date** hinzugefügt, bevor **xs:date** zurückgegeben wird. Der Typ `xs:date` hat das Format `JJJJ-MM-TT`.

▼ add-years-to-date [altova:]

altova:add-years-to-date(Date *als xs:date*, **Years** *als xs:integer*) als **xs:date** XP3.1 XQ3.1
Fügt eine Zeitdauer in Jahren zu einem Datumswert hinzu. Beim zweiten Argument handelt es sich um die Anzahl der Jahre, die zu dem im ersten Argument angegebenen `xs:date` Wert hinzugefügt werden sollen. Das Ergebnis ist vom Typ `xs:date`.

Beispiele

- **altova:add-years-to-date**(xs:date("2014-01-15"), 10) gibt 2024-01-15 zurück
- **altova:add-years-to-date**(xs:date("2014-01-15"), -4) gibt 2010-01-15 zurück

▼ add-months-to-date [altova:]

altova:add-months-to-date(Date *als xs:date*, **Months** *als xs:integer*) als **xs:date** XP3.1 XQ3.1
Fügt eine Zeitdauer in Monaten zu einem Datumswert hinzu. Beim zweiten Argument handelt es sich um die Anzahl der Monate, die zu dem im ersten Argument angegebenen `xs:date` Wert hinzugefügt werden sollen. Das Ergebnis ist vom Typ `xs:date`.

Beispiele

- **altova:add-months-to-date**(xs:date("2014-01-15"), 10) gibt 2014-11-15 zurück
- **altova:add-months-to-date**(xs:date("2014-01-15"), -2) gibt 2013-11-15 zurück

▼ add-days-to-date [altova:]

altova:add-days-to-date(Date *als xs:date*, **Days** *als xs:integer*) als **xs:date** XP3.1 XQ3.1
Fügt eine Zeitdauer in Tagen zu einem Datumswert hinzu. Beim zweiten Argument handelt es sich um die Anzahl der Tage, die zu dem im ersten Argument angegebenen `xs:date` Wert hinzugefügt werden sollen. Das Ergebnis ist vom Typ `xs:date`.

Beispiele

- **altova:add-days-to-date**(xs:date("2014-01-15"), 10) gibt 2014-01-25 zurück
- **altova:add-days-to-date**(xs:date("2014-01-15"), -8) gibt 2014-01-07 zurück

[Nach oben 439]

Formatieren und Abrufen einer Zeitdauer XP3.1 XQ3.1

Mit diesen Funktionen wird ein Input **xs:duration**- oder **xs:string**-Wert geparst und ein **xs:string**- bzw. **xs:duration**-Wert zurückgegeben.

▼ format-duration [altova:]

altova:format-duration(Duration *als xs:duration*, **Picture** *als xs:string*) als **xs:string** XP3.1 XQ3.1
Formatiert eine Zeitdauer, die als erstes Argument bereitgestellt wird, gemäß einem Muster-String, der als zweites Argument bereitgestellt wird. Die Ausgabe ist ein Textstring, der dem Muster-String entsprechend

formatiert ist.

Beispiele

- `altova:format-duration(xs:duration("P2DT2H53M11.7S"), "Days:[D01] Hours:[H01] Minutes:[m01] Seconds:[s01] Fractions:[f0]")` gibt `"Days:02 Hours:02 Minutes:53 Seconds:11 Fractions:7"` gibt
- `altova:format-duration(xs:duration("P3M2DT2H53M11.7S"), "Months:[M01] Days:[D01] Hours:[H01] Minutes:[m01]")` gibt `"Months:03 Days:02 Hours:02 Minutes:53"` gibt

parse-duration [altova:]

altova:parse-duration(InputString *als xs:string*, **Picture** *als xs:string*) als **xs:duration** XP3.1 XQ3.1

Erhält einen Pattern-String als erstes Argument und eine Muster-String als zweites Argument. Der Input-Strings wird auf Basis des Muster-Strings geparst und ein `xs:duration` wird zurückgegeben.

Beispiele

- `altova:parse-duration("Days:02 Hours:02 Minutes:53 Seconds:11 Fractions:7", "Days:[D01] Hours:[H01] Minutes:[m01] Seconds:[s01] Fractions:[f0]")` gibt `"P2DT2H53M11.7S"` zurück
- `altova:parse-duration("Months:03 Days:02 Hours:02 Minutes:53 Seconds:11 Fractions:7", "Months:[M01] Days:[D01] Hours:[H01] Minutes:[m01]")` gibt `"P3M2DT2H53M"` zurück

[Nach oben 439]

Hinzufügen einer Zeitdauer zu xs:time XP3.1 XQ3.1

Diese Funktionen fügen einen Zeitdauerwert zu **xs:time** hinzu und geben **xs:time** zurück. Der Typ `xs:time` entspricht in seiner lexikalischen Form `hh:mm:ss.sss`. Eine optionale Zeitzone kann angehängt werden. Der Buchstabe `Z` steht für Coordinated Universal Time (UTC). Alle anderen Zeitzonen werden in Form des Zeitunterschieds zur UTC im Format `+hh:mm`, oder `-hh:mm` dargestellt. Wenn kein Wert für die Zeitzone vorhanden ist, wird sie als unbekannt und nicht als UTC angenommen.

add-hours-to-time [altova:]

altova:add-hours-to-time(Time *als xs:time*, **Hours** *als xs:integer*) als **xs:time** XP3.1 XQ3.1

Fügt eine Zeitdauer in Stunden zu einem Uhrzeitwert hinzu. Beim zweiten Argument handelt es sich um die Anzahl der Stunden, die zu dem im ersten Argument angegebenen `xs:time` Wert hinzugefügt werden sollen. Das Ergebnis ist vom Typ `xs:time`.

Beispiele

- `altova:add-hours-to-time(xs:time("11:00:00"), 10)` gibt `21:00:00` zurück
- `altova:add-hours-to-time(xs:time("11:00:00"), -7)` gibt `04:00:00` zurück

add-minutes-to-time [altova:]

altova:add-minutes-to-time(Time *als xs:time*, **Minutes** *als xs:integer*) als **xs:time** XP3.1 XQ3.1

Fügt eine Zeitdauer in Minuten zu einem `xs:time` Wert hinzu. Beim zweiten Argument handelt es sich um die Anzahl der Minuten, die zu dem im ersten Argument angegebenen `xs:time` Wert hinzugefügt werden

sollen. Das Ergebnis ist vom Typ `xs:time`.

Beispiele

- **`altova:add-minutes-to-time`**`(xs:time("14:10:00"), 45)` gibt `14:55:00` zurück
- **`altova:add-minutes-to-time`**`(xs:time("14:10:00"), -5)` gibt `14:05:00` zurück

add-seconds-to-time [altova:]

`altova:add-seconds-to-time(Time` *`as xs:time`*`, `**`Minutes`** *`als xs:integer`*`)` als **`xs:time`** **XP3.1** **XQ3.1**

Fügt eine Zeitdauer in Sekunden zu einem Uhrzeitwert hinzu. Beim zweiten Argument handelt es sich um die Anzahl der Sekunden, die zu dem im ersten Argument angegebenen `xs:time` Wert hinzugefügt werden sollen. Das Ergebnis ist vom Typ `xs:time`. Die Seconds Komponenten kann sich im Bereich von `0` bis `59.999` befinden.

Beispiele

- **`altova:add-seconds-to-time`**`(xs:time("14:00:00"), 20)` gibt `14:00:20` zurück
- **`altova:add-seconds-to-time`**`(xs:time("14:00:00"), 20.895)` gibt `14:00:20.895` zurück

[Nach oben 439]

Entfernen der Zeitzone aus date/time-Datentypen **XP3.1** **XQ3.1**

Diese Funktionen entfernen die Zeitzone aus den aktuellen **`xs:dateTime`**, **`xs:date`** bzw. **`xs:time`** Werten. Beachten Sie, dass im Unterschied zu `xs:dateTime` bei `xs:dateTimeStamp` die Zeitzone erforderlich ist (während sie im ersteren Fall optional ist). Das Format eines `xs:dateTimeStamp` Werts lautet daher: `JJJJ-MM-TTZhh:mm:ss.sss±hh:mm`. oder `JJJJ-MM-TTZhh:mm:ss.sssZ`. Wenn das Datum und die Uhrzeit von der Systemuhr als `xs:dateTimeStamp` ausgelesen wird, können Sie die Zeitzone, falls erforderlich, mit der Funktion `current-dateTime-no-TZ()` entfernen.

current-date-no-TZ [altova:]

`altova:current-date-no-TZ()` als **`xs:date`** **XP3.1** **XQ3.1**

Die Funktion hat kein Argument. Sie entfernt die Zeitzone aus dem `current-date()` Wert (welcher das aktuelle Datum laut Systemuhr ist) und gibt einen `xs:date` Wert zurück.

Beispiele

Wenn das aktuelle Datum **`2014-01-15+01:00`** lautet:

- **`altova:current-date-no-TZ`**`()` gibt `2014-01-15` zurück

current-dateTime-no-TZ [altova:]

`altova:current-dateTime-no-TZ()` als **`xs:dateTime`** **XP3.1** **XQ3.1**

Die Funktion hat kein Argument. Sie entfernt die Zeitzone aus dem `current-dateTime()` Wert (welcher das aktuelle Datum und die aktuelle Uhrzeit laut Systemuhr ist) und gibt einen `xs:dateTime` Wert zurück.

Beispiele

Wenn der aktuelle Datums- und Uhrzeitwert **`2014-01-15T14:00:00+01:00`** lautet:

- **altova:current-dateTime-no-TZ()** gibt 2014-01-15T14:00:00 zurück

▼ current-time-no-TZ [altova:]

altova:current-time-no-TZ() als **xs:time** XP3.1 XQ3.1
Die Funktion hat kein Argument. Sie entfernt die Zeitzone aus dem current-time() Wert (welcher die aktuelle Uhrzeit laut Systemuhr ist) und gibt einen xs:time Wert zurück.

Beispiele

Wenn der aktuelle Uhrzeitwert **14:00:00+01:00** lautet:

- **altova:current-time-no-TZ()** gibt 14:00:00 zurück

▼ date-no-TZ [altova:]

altova:date-no-TZ(InputDate *as xs:date*) als **xs:date** XP3.1 XQ3.1
Diese Funktion verwendet ein xs:date Argument, entfernt den Zeitzonenteil daraus und gibt einen xs:date Wert zurück. Beachten Sie, dass das Datum nicht geändert wird..

Beispiele

- **altova:date-no-TZ**(xs:date("2014-01-15+01:00")) gibt 2014-01-15 zurück

▼ dateTime-no-TZ [altova:]

altova:dateTime-no-TZ(InputDateTime *als xs:dateTime*) als **xs:dateTime** XP3.1 XQ3.1
Diese Funktion verwendet ein xs:dateTime Argument, entfernt den Zeitzonenteil daraus und gibt einen xs:dateTime Wert zurück. Beachten Sie, dass weder Datum noch Uhrzeit geändert werden.

Beispiele

- **altova:dateTime-no-TZ**(xs:date("2014-01-15T14:00:00+01:00")) gibt 2014-01-15T14:00:00 zurück

▼ time-no-TZ [altova:]

altova:time-no-TZ(InputTime *als xs:time*) als **xs:time** XP3.1 XQ3.1
Diese Funktion verwendet ein xs:time Argument, entfernt den Zeitzonenteil daraus und gibt einen xs:time Wert zurück. Beachten Sie, dass die Uhrzeit nicht geändert wird.

Beispiele

- **altova:time-no-TZ**(xs:time("14:00:00+01:00")) gibt 14:00:00 zurück

[Nach oben 439]

Rückgabe der Anzahl von Tagen, Stunden, Minuten, Sekunden anhand einer Zeitdauer XP3.1 XQ3.1

Diese Funktionen geben die Anzahl der Tage in einem Monat bzw. die Anzahl der Stunden, Minuten und Sekunden anhand einer Zeitdauer zurück.

days-in-month [altova:]

altova:days-in-month(Year *als xs:integer*, **Month** *als xs:integer*) als **xs:integer** **XP3.1** **XQ3.1**
Gibt die Anzahl der Tage im angegebenen Monat zurück. Der Monat wird mit Hilfe der Argumente `Year` und `Month` angegeben.

Beispiele

- `altova:days-in-month(2018, 10)` gibt `31` zurück
- `altova:days-in-month(2018, 2)` gibt `28` zurück
- `altova:days-in-month(2020, 2)` gibt `29` zurück

hours-from-dayTimeDuration-accumulated

altova:hours-from-dayTimeDuration-accumulated(DayAndTime *als xs:duration*) als **xs:integer** **XP3.1** **XQ3.1**
Gibt die Gesamtzahl der Stunden in der durch das Argument `DayAndTime` bereitgestellten Zeitdauer (vom Typ `xs:duration`) zurück. Die Stunden in den Komponenten `Day` und `Time` werden addiert, um ein Ergebnis in Form einer Ganzzahl zu erhalten. Nur für volle 60 Minuten wird eine neue Stunde berechnet. Negative Zeitdauerergebnisse ergeben einen negativen Stundenwert.

Beispiele

- `altova:hours-from-dayTimeDuration-accumulated(xs:duration("P5D"))` gibt `120` zurück, d.h. die Gesamtanzahl der Stunden in 5 Tagen.
- `altova:hours-from-dayTimeDuration-accumulated(xs:duration("P5DT2H"))` gibt `122` zurück, d.h. die Gesamtzahl der Stunden in 5 Tagen plus 2 Stunden.
- `altova:hours-from-dayTimeDuration-accumulated(xs:duration("P5DT2H60M"))` gibt `123` zurück, d.h. die Gesamtzahl der Stunden in 5 Tagen plus 2 Stunden und 60 Minuten.
- `altova:hours-from-dayTimeDuration-accumulated(xs:duration("P5DT2H119M"))` gibt `123` zurück, d.h. die Gesamtzahl der Stunden in 5 Tagen plus 2 Stunden und 119 Minuten.
- `altova:hours-from-dayTimeDuration-accumulated(xs:duration("P5DT2H120M"))` gibt `124` zurück, d.h. die Gesamtzahl der Stunden in 5 Tagen plus 2 Stunden und 120 Minuten.
- `altova:hours-from-dayTimeDuration-accumulated(xs:duration("-P5DT2H"))` gibt `-122` zurück.

minutes-from-dayTimeDuration-accumulated

altova:minutes-from-dayTimeDuration-accumulated(DayAndTime *als xs:duration*) als **xs:integer** **XP3.1** **XQ3.1**
Gibt die Gesamtzahl der Minuten in der durch das Argument `DayAndTime` bereitgestellten Zeitdauer (vom Typ `xs:duration`) zurück. Die Minuten in den Komponenten `Day` und `Time` werden addiert, um ein Ergebnis in Form einer Ganzzahl zu erhalten. Negative Zeitdauerergebnisse ergeben einen negativen Minutenwert.

Beispiele

- `altova:minutes-from-dayTimeDuration-accumulated(xs:duration("PT60M"))` gibt `60` zurück.
- `altova:minutes-from-dayTimeDuration-accumulated(xs:duration("PT1H"))` gibt `60` zurück, d.h. die Gesamtzahl der Minuten in 1 Stunde.
- `altova:minutes-from-dayTimeDuration-accumulated(xs:duration("PT1H40M")` gibt `100` zurück.

- **altova:minutes-from-dayTimeDuration-accumulated**(xs:duration("P1D")) gibt 1440 zurück, d.h. die Gesamtzahl der Minuten an einem Tag.
- **altova:minutes-from-dayTimeDuration-accumulated**(xs:duration("-P1DT60M")) gibt -1500 zurück.

▼ seconds-from-dayTimeDuration-accumulated

altova:seconds-from-dayTimeDuration-accumulated(DayAndTime *als xs:duration*) als **xs:integer** XP3.1 XQ3.1

Gibt die Gesamtzahl der Sekunden in der durch das Argument DayAndTime bereitgestellten Zeitdauer (vom Typ xs:duration) zurück. Die Sekunden in den Komponenten Day und Time werden addiert, um ein Ergebnis in Form einer Ganzzahl zu erhalten. Negative Zeitdauerergebnisse ergeben einen negativen Sekundenwert.

▣ *Examples*

- **altova:seconds-from-dayTimeDuration-accumulated**(xs:duration("PT1M")) gibt 60 zurück, d.h. die Gesamtzahl der Sekunden in 1 Minute.
- **altova:seconds-from-dayTimeDuration-accumulated**(xs:duration("PT1H")) gibt 3600 zurück, d.h. die Gesamtzahl der Sekunden in 1 Stunde.
- **altova:seconds-from-dayTimeDuration-accumulated**(xs:duration("PT1H2M")) gibt 3720 zurück.
- **altova:seconds-from-dayTimeDuration-accumulated**(xs:duration("P1D"))gibt 86400 zurück, d.h. die Gesamtzahl der Sekunden an 1 Tag.
- **altova:seconds-from-dayTimeDuration-accumulated**(xs:duration("-P1DT1M")) gibt -86460 zurück.

Rückgabe des Wochentages anhand von xs:dateTime oder xs:date XP3.1 XQ3.1

Diese Funktionen geben anhand des xs:dateTime oder xs:date Werts den Wochentag in Form einer Ganzzahl zurück. Die Tage der Woche sind (im amerikanischen Format) von 1 bis 7 nummeriert, wobei Sonntag=1. Im europäischen Format beginnt die Woche am Montag (=1). Das amerikanische Format, in dem Sonntag=1, kann mittels der Ganzzahl 0 definiert werden, wenn das Format mittels einer Ganzzahl angegeben werden kann.

▼ weekday-from-dateTime [altova:]

altova:weekday-from-dateTime(DateTime *als xs:dateTime*) **als xs:integer** XP3.1 XQ3.1

Erhält ein Datum mit einer Uhrzeit als einziges Argument und gibt den Tag der Woche dieses Datums in Form einer Ganzzahl zurück. Die Wochentage sind beginnend mit Sonntag=1 nummeriert. Wenn das europäische Format benötigt wird (wo Montag=1), verwenden Sie die andere Signatur dieser Funktion (*siehe nächste Signatur unten*).

▣ *Beispiele*

- **altova:weekday-from-dateTime**(xs:dateTime("2014-02-03T09:00:00")) gibt 2 zurück, wobei 2 für Montag steht.

altova:weekday-from-dateTime(DateTime *als xs:dateTime*, **Format** *als xs:integer*) **als xs:integer** XP3.1 XQ3.1

Erhält ein Datum mit einer Uhrzeit als erstes Argument und gibt den Tag der Woche dieses Datums in

Form einer Ganzzahl zurück. Wenn das zweite (Integer)-Argument `0` ist, werden die Wochentage beginnend mit `Sonntag=1` von `1` bis `7` nummeriert. Wenn das zweite Argument eine andere Ganzzahl als `0` ist, so ist `Montag=1`. Wenn es kein zweites Argument gibt, wird die Funktion gelesen, als ob sie die andere Signatur dieser Funktion hätte (*siehe vorherige Signatur*).

Beispiele

- **altova:weekday-from-dateTime**(xs:dateTime("2014-02-03T09:00:00"), 1) gibt 1 zurück, wobei 1 für Montag steht
- **altova:weekday-from-dateTime**(xs:dateTime("2014-02-03T09:00:00"), 4) gibt 1 zurück, wobei 1 für Montag steht
- **altova:weekday-from-dateTime**(xs:dateTime("2014-02-03T09:00:00"), 0) gibt 2 zurück, wobei 2 für Montag steht.

weekday-from-date [altova:]

altova:weekday-from-date(Date *als xs:date*) **als xs:integer** XP3.1 XQ3.1

Erhält ein Datum als einziges Argument und gibt den Tag der Woche dieses Datums in Form einer Ganzzahl zurück. Die Wochentage sind beginnend mit `Sonntag=1` nummeriert. Wenn das europäische Format benötigt wird (wo `Montag=1`), verwenden Sie die andere Signatur dieser Funktion (*siehe nächste Signatur unten*).

Beispiele

- **altova:weekday-from-date**(xs:date("2014-02-03+01:00")) gibt 2 zurück, d.h. dies wäre ein Montag.

altova:weekday-from-date(Date *als xs:date*, **Format** *als xs:integer*) **als xs:integer** XP3.1 XQ3.1

Erhält ein Datum als erstes Argument und gibt den Tag der Woche dieses Datums in Form einer Ganzzahl zurück. Wenn das zweite Argument (`Format) 0` ist, werden die Wochentage beginnend mit `Sonntag=1` von `1` bis `7` nummeriert. Wenn das zweite Argument eine andere Ganzzahl als `0` ist, so ist `Montag=1`. Wenn es kein zweites Argument gibt, wird die Funktion gelesen, als ob sie die andere Signatur dieser Funktion hätte (*siehe vorherige Signatur*).

Beispiele

- **altova:weekday-from-date**(xs:date("2014-02-03"), 1) gibt 1 zurück, wobei 1 für Montag steht
- **altova:weekday-from-date**(xs:date("2014-02-03"), 4) gibt 1 zurück, wobei 1 für Montag steht
- **altova:weekday-from-date**(xs:date("2014-02-03"), 0) gibt 2 zurück, wobei 2 für Montag steht.

[Nach oben 439]

Rückgabe der Wochennummer anhand von xs:dateTime oder xs:date XP2 XQ1 XP3.1 XQ3.1

Diese Funktionen geben anhand von `xs:dateTime` oder `xs:date` die Wochennummer als Ganzzahl zurück. Die Wochennummer steht in den Kalenderformaten US, ISO/European und Islamic zur Verfügung. Die Wochennummerierung unterscheidet sich in diesen Kalenderformaten, da die Woche in diesen Formaten an unterschiedlichen Tagen beginnt (Im Format US am Sonntag, im Format ISO/European am Montag und im Format Islamic am Samstag).

▼ weeknumber-from-date [altova:]

altova:weeknumber-from-date(Date *als xs:date*, **Calendar** *als xs:integer*) als **xs:integer** XP2 XQ1 XP3.1 XQ3.1

Gibt die Wochennummer des bereitgestellten **Date** Arguments als Ganzzahl zurück. Das zweite Argument (**Calendar**) definiert das zu verwendende Kalendersystem.
Unterstützte **Calendar** Werte sind:

- **0 = US-Kalender** (Woche beginnt am Sonntag)
- **1 = ISO-Standard, Europäischer Kalender** (Woche beginnt am Montag)
- **2 = Islamischer Kalender** (Woche beginnt am Samstag)

Der Standardwert ist 0.

▣ *Beispiele*

- **altova:weeknumber-from-date**(xs:date("2014-03-23"), 0) gibt 13 zurück
- **altova:weeknumber-from-date**(xs:date("2014-03-23"), 1) gibt 12 zurück
- **altova:weeknumber-from-date**(xs:date("2014-03-23"), 2) gibt 13 zurück
- **altova:weeknumber-from-date**(xs:date("2014-03-23")) gibt 13 zurück

Der Tag des Datums in den obigen Beispielen (2014-03-23) ist ein Sonntag. Daher ist der US- und der islamische Kalender dem europäischen Kalender an diesem Tag eine Woche voraus.

▼ weeknumber-from-dateTime [altova:]

altova:weeknumber-from-dateTime(DateTime *als xs:dateTime*, **Calendar** *als xs:integer*) **als xs:integer** XP2 XQ1 XP3.1 XQ3.1

Gibt die Wochennummer des bereitgestellten **DateTime** Arguments als Ganzzahl zurück. Das zweite Argument (**Calendar**) definiert das zu verwendende Kalendersystem.
Unterstützte **Calendar** Werte sind:

- **0 = US-Kalender** (*Woche beginnt am Sonntag*)
- **1 = ISO-Standard, Europäischer Kalender** (*Woche beginnt am Montag*)
- **2 = Islamischer Kalender** (*Woche beginnt am Samstag*)

Der Standardwert ist 0.

▣ *Beispiele*

- **altova:weeknumber-from-dateTime**(xs:dateTime("2014-03-23T00:00:00"), 0) gibt 13 zurück
- **altova:weeknumber-from-dateTime**(xs:dateTime("2014-03-23T00:00:00"), 1) gibt 12 zurück
- **altova:weeknumber-from-dateTime**(xs:dateTime("2014-03-23T00:00:00"), 2) gibt 13 zurück
- **altova:weeknumber-from-dateTime**(xs:dateTime("2014-03-23T00:00:00")) gibt 13 zurück

Der Tag des Datums- und Uhrzeitwerts in den obigen Beispielen (2014-03-23T00:00:00) ist ein

Sonntag. Daher ist der US- und der islamische Kalender dem europäischen Kalender an diesem Tag eine Woche voraus.

[Nach oben 439]

Erstellen des Datums-, Uhrzeit- oder Zeitdauer-Datentyps anhand der lexikalischen Komponenten der einzelnen Typen XP3.1 XQ3.1

Die Funktionen erhalten die lexikalischen Komponenten des `xs:date`, `xs:time` oder `xs:duration`-Datentyps als Input-Argumente und kombinieren diese zum entsprechenden Datentyp.

▼ build-date [altova:]

altova:build-date(Year *als xs:integer*, **Month** *als xs:integer*, **Date** *als xs:integer*) als **xs:date** XP3.1 XQ3.1

Das erste, zweite und dritte Argument steht für das Jahr, bzw. den Monat bzw. das Datum. Sie werden zu einem Wert vom Typ `xs:date` kombiniert. Die Werte der Ganzzahlen müssen sich innerhalb des korrekten Bereichs dieses jeweiligen Datumsteils befinden. So sollte z.B. das zweite Argument nicht größer als `12` sein.

▣ *Beispiele*

- **altova:build-date**(2014, 2, 03) gibt 2014-02-03 zurück

▼ build-time [altova:]

altova:build-time(Hours *als xs:integer*, **Minutes** *als xs:integer*, **Seconds** *als xs:integer*) als **xs:time** XP3.1 XQ3.1

Das erste, zweite und dritte Argument steht für die Stunde (`0` bis `23`), bzw. die Minuten (`0` bis `59`) bzw. die Sekunden (`0` bis `59`). Sie werden zu einem Wert von Typ `xs:time` kombiniert. Die Werte der Ganzzahlen müssen sich innerhalb des korrekten Bereichs dieses jeweiligen Uhrzeitteils befinden. So sollte z.B. der zweite Parameter nicht größer als `59` sein. Um eine Zeitzone zum Wert hinzuzufügen, verwenden Sie die andere Signatur der Funktion (*siehe nächste Signatur*).

▣ *Beispiele*

- **altova:build-time**(23, 4, 57) gibt 23:04:57 zurück

altova:build-time(Hours *als xs:integer*, **Minutes** *als xs:integer*, **Seconds** *als xs:integer*, **TimeZone** *als xs:string*) als **xs:time** XP3.1 XQ3.1

Das erste, zweite und dritte Argument steht für die Stunde (`0` bis `23`), bzw. die Minuten (`0` bis `59`) bzw. die Sekunden (`0` bis `59`). Das vierte Argument ist ein String, der den Zeitzonenteil des Werts liefert. Die vier Argumente werden zu einem Wert vom Typ `xs:time` kombiniert. Die Werte der Ganzzahlen müssen sich innerhalb des korrekten Bereichs dieses jeweiligen Uhrzeitteils befinden. So sollte z.B. das zweite Argument (Minuten) nicht größer als `59` sein.

▣ *Beispiele*

- **altova:build-time**(23, 4, 57, '+1') gibt 23:04:57+01:00 zurück

▼ build-duration [altova:]

altova:build-duration(Years *als xs:integer*, **Months** *als xs:integer*) als **xs:yearMonthDuration** XP3.1 XQ3.1

Setzt aus zwei Argumenten einen Wert vom Typ `xs:yearMonthDuration` zusammen. Das erste Argument liefert den Jahr-Teil des Zeitdauerwerts, während das zweite Argument den Monat-Teil liefert. Wenn der zweite Parameter (Monate) größer oder gleich `12` ist, so wird die Ganzzahl durch 12 dividiert. Der Quotient wird zum ersten Argument hinzugefügt, um den Jahr-Teil des Zeitdauerwerts zu liefern, während der Rest (der Division) den Monat-Teil liefert. Eine Beschreibung zur Erstellung einer Zeitdauer vom Typ `xs:dayTimeDuration` finden Sie in der nächsten Signatur.

Beispiele

- **altova:build-duration**(2, 10) gibt `P2Y10M` zurück
- **altova:build-duration**(14, 27) gibt `P16Y3M` zurück
- **altova:build-duration**(2, 24) gibt `P4Y` zurück

altova:build-duration(Days *als xs:integer*, **Hours** *als xs:integer*, **Minutes** *als xs:integer*, **Seconds** *als xs:integer*) als **xs:dayTimeDuration** XP3.1 XQ3.1

Kombiniert vier Argumente zu einem Wert vom Typ `xs:dayTimeDuration`. Das erste Argument liefert den Tage-Teil, das zweite die Stunden, das dritte die Minuten und das vierte die Sekunden des Zeitdauerwerts. Die einzelnen Uhrzeitparameter werden in den entsprechenden Wert für die nächsthöhere Einheit konvertiert und das Ergebnis wird zur Berechnung der Gesamtdauer weitergegeben. So werden z.B. 72 Sekunden in `1M(inute)12S(ekunden)` konvertiert. Dieser Wert wird zur Berechnung der Gesamtdauer weitergegeben. Um eine Zeitdauer vom Typ `xs:yearMonthDuration` zu berechnen, verwenden Sie die vorherige Signatur.

Beispiele

- **altova:build-duration**(2, 10, 3, 56) gibt `P2DT10H3M56S` zurück
- **altova:build-duration**(1, 0, 100, 0) gibt `P1DT1H40M` zurück
- **altova:build-duration**(1, 0, 0, 3600) gibt `P1DT1H` zurück

[Nach oben 439]

Konstruieren von Datum, Datum und Uhrzeit und Zeit-Datentypen anhand des String-Input XP2 XQ1 XP3.1 XQ3.1

Diese Funktionen erhalten Strings als Argumente und konstruieren anhand dieser die Datentypen `xs:date`, `xs:dateTime` oder `xs:time`. Der String wird anhand eines bereitgestellten Pattern-Arguments nach Komponenten des Datentyps analysiert.

parse-date [altova:]

altova:parse-date(Date *als xs:string*, **DatePattern** *als xs:string*) als **xs:date** XP2 XQ1 XP3.1 XQ3.1

Gibt den Input-String **Date** als **xs:date** Wert zurück. Das zweite Argument **DatePattern** definiert das Pattern (die Komponentensequenz) des Input-String. **DatePattern** wird durch die unten aufgelisteten Komponenten-Specifier beschrieben. Als Komponententrennzeichen kann jedes beliebige Zeichen verwendet werden. Siehe Beispiele unten.

D	Datum
M	Monat
Y	Jahr

Das Pattern in **DatePattern** muss mit dem Pattern in **Date** übereinstimmen. Da die Ausgabe vom Typ **xs:date** ist, hat sie immer das lexikalische Format **YYYY-MM-DD**.

Beispiele

- **altova:parse-date**(xs:string("09-12-2014"), "[D]-[M]-[Y]") gibt 2014-12-09 zurück
- **altova:parse-date**(xs:string("09-12-2014"), "[M]-[D]-[Y]") gibt 2014-09-12 zurück
- **altova:parse-date**("06/03/2014", "[M]/[D]/[Y]") gibt 2014-06-03 zurück
- **altova:parse-date**("06 03 2014", "[M] [D] [Y]") gibt 2014-06-03 zurück
- **altova:parse-date**("6 3 2014", "[M] [D] [Y]") gibt 2014-06-03 zurück

parse-dateTime [altova:]

altova:parse-dateTime(DateTime *als xs:string*, **DateTimePattern** *als xs:string*) als **xs:dateTime** **XP2** **XQ1** **XP3.1** **XQ3.1**

Gibt den Input-String **DateTime** als **xs:dateTime** Wert zurück. Das zweite Argument **DateTimePattern** definiert das Pattern (die Komponentensequenz) des Input-String. **DateTimePattern** wird durch die unten aufgelisteten Komponenten-Specifier beschrieben. Als Komponententrennzeichen kann jedes beliebige Zeichen verwendet werden. Siehe Beispiele unten.

D	Datum
M	Monat
Y	Jahr
H	Stunde
m	Minuten
s	Sekunden

Das Pattern in **DateTimePattern** muss mit dem Pattern in **DateTime** übereinstimmen. Da die Ausgabe vom Typ **xs:dateTime** ist, hat sie immer das lexikalische Format **YYYY-MM-DDTHH:mm:ss**.

Beispiele

- **altova:parse-dateTime**(xs:string("09-12-2014 13:56:24"), "[M]-[D]-[Y] [H]:[m]:[s]") gibt 2014-09-12T13:56:24 zurück
- **altova:parse-dateTime**("time=13:56:24; date=09-12-2014", "time=[H]:[m]:[s]; date=[D]-[M]-[Y]") gibt 2014-12-09T13:56:24 zurück

parse-time [altova:]

altova:parse-time(Time *als xs:string*, **TimePattern** *als xs:string*) als **xs:time** **XP2** **XQ1** **XP3.1** **XQ3.1**

Gibt den Input-String **Time** als **xs:time** Wert zurück. Das zweite Argument **TimePattern** definiert das Pattern (die Komponentensequenz) des Input-String. **TimePattern** wird durch die unten aufgelisteten Komponenten-Specifier beschrieben. Als Komponententrennzeichen kann jedes beliebige Zeichen verwendet werden. Siehe Beispiele unten.

H	Stunde
m	Minuten
s	Sekunden

Das Pattern in **TimePattern** muss mit dem Pattern in **Time** übereinstimmen. Da die Ausgabe vom Typ **xs:Time** ist, hat sie immer das lexikalische Format **HH:mm:ss**.

Beispiele

- **altova:parse-time**(xs:string("13:56:24"), "[H]:[m]:[s]") gibt 13:56:24 zurück
- **altova:parse-time**("13-56-24", "[H]-[m]") gibt 13:56:00 zurück
- **altova:parse-time**("time=13h56m24s", "time=[H]h[m]m[s]s") gibt 13:56:24 zurück
- **altova:parse-time**("time=24s56m13h", "time=[s]s[m]m[H]h") gibt 13:56:24 zurück

[Nach oben 439]

Funktionen zur Berechnung des Alters XP3.1 XQ3.1

Diese Funktionen geben das Alter berechnet (i) anhand von einem Input-Argument und dem aktuellen Datum oder (ii) anhand zweier Input-Argumentdaten zurück. Die Funktion **altova:age** gibt das Alter in Jahren zurück, die Funktion **altova:age-details** gibt das Alter als Sequenz von drei Ganzzahlen zurück, die die Jahre, Monate und Tage des Alters angeben.

age [altova:]

altova:age(StartDate *als xs:date*) als **xs:integer** XP3.1 XQ3.1
Gibt eine Ganzzahl zurück, die das Alter eines Objekts in *Jahren* angibt. Berechnet wird das Alter anhand des durch das Argument gelieferten Startdatums endend mit dem aktuellen Datum (laut Systemuhr). Wenn das Input-Argument eines Datums größer oder gleich einem Jahr in der Zukunft ist, ist der Rückgabewert negativ.

Beispiele

Wenn das aktuelle Datum **2014-01-15** lautet:

- **altova:age**(xs:date("2013-01-15")) gibt 1 zurück
- **altova:age**(xs:date("2013-01-16")) gibt 0 zurück
- **altova:age**(xs:date("2015-01-15")) gibt -1 zurück
- **altova:age**(xs:date("2015-01-14")) gibt 0 zurück

altova:age(StartDate *als xs:date*, **EndDate** *als xs:date*) als **xs:integer** XP3.1 XQ3.1
Gibt eine Ganzzahl zurück, die das Alter eines Objekts in *Jahren* angibt. Berechnet wird das Alter anhand des durch das erste Argument gelieferten Startdatums endend mit dem als zweites Datum gelieferten Enddatum. Wenn das erste Argument ein Jahr oder mehr nach dem zweiten Argument liegt, ist der Rückgabewert negativ.

Beispiele

Wenn das aktuelle Datum **2014-01-15** lautet:

- **altova:age**(xs:date("2000-01-15"), xs:date("2010-01-15")) gibt 10 zurück
- **altova:age**(xs:date("2000-01-15"), current-date()) gibt 14 zurück, wenn das aktuelle Datum 2014-01-15 ist
- **altova:age**(xs:date("2014-01-15"), xs:date("2010-01-15")) gibt -4 zurück

▼ age-details [altova:]

altova:age-details(InputDate *als xs:date*) als (**xs:integer)*** **XP3.1** **XQ3.1**
Gibt drei Ganzzahlen zurück. Dabei handelt es sich um die Jahre, Monate bzw. Tage zwischen dem als Argument angegebenen Datum und dem aktuellen Datum (laut Systemuhr). Die Summe der zurückgegebenen `years+months+days` gibt zusammen die Gesamtzeitdifferenz zwischen den beiden Datumswerten (dem Input-Datum und dem aktuellen Datum) an. Das Input-Datum hat eventuell einen Wert, der vor oder nach dem aktuellen Datum liegt, doch wird dies nicht aus dem Vorzeichen der Rückgabewerte ersichtlich; die Rückgabewerte sind immer positiv.

Beispiele

Wenn das aktuelle Datum **2014-01-15** lautet:

- `altova:age-details(xs:date("2014-01-16"))` gibt `(0 0 1)` zurück
- `altova:age-details(xs:date("2014-01-14"))` gibt `(0 0 1)` zurück
- `altova:age-details(xs:date("2013-01-16"))` gibt `(1 0 1)` zurück
- `altova:age-details(current-date())` gibt `(0 0 0)` zurück

altova:age-details(Date-1 *als xs:date*, **Date-2** *als xs:date*) als (**xs:integer)*** **XP3.1** **XQ3.1**
Gibt drei Ganzzahlen zurück. Dabei handelt es sich um die Jahre, Monate bzw. Tage zwischen den beiden Argumentdaten. Die Summe der zurückgegebenen `years+months+days` gibt zusammen die Gesamtzeitdifferenz zwischen den beiden Input-Datumswerten an. Es ist unerheblich, ob das frühere oder spätere Datum als erstes Argument angegeben wird. Die Rückgabewerte geben nicht an, ob das Input-Datum vor oder nach dem aktuellen Datum liegt. Die Rückgabewerte sind immer positiv.

Beispiele

- `altova:age-details(xs:date("2014-01-16"), xs:date("2014-01-15"))` gibt `(0 0 1)` zurück
- `altova:age-details(xs:date("2014-01-15"), xs:date("2014-01-16"))` gibt `(0 0 1)` zurück

[Nach oben 439]

Epochen-Zeit (Unix-Zeit)-Funktionen **XP3.1** **XQ3.1**

Die Epochenzeit ist ein auf Unix-Systemen verwendetes Zeitsystem. Darin wird jeder Zeitpunkt als Anzahl der Sekunden seit 00:00:00 UTC des 1. Januar 1970 definiert. Diese Epochenzeitfunktionen konvertieren **xs:dateTime**-Werte in Epochenzeitwerte und umgekehrt.

▼ dateTime-from-epoch [altova:]

altova:dateTime-from-epoch(Epoch *als xs:decimal* **als xs:dateTime** **XP3.1** **XQ3.1**
Die Epochenzeit ist ein auf Unix-Systemen verwendetes Zeitsystem. Darin wird jeder Zeitpunkt als Anzahl der Sekunden seit 00:00:00 UTC des 1. Januar 1970 definiert. Die Funktion **dateTime-from-epoch** gibt das **xs:dateTime**-Äquivalent einer Epochenzeit zurück, passt die lokale Zeitzone an und inkludiert die Zeitzoneninformation im Ergebnis.

Die Funktion erhält ein `xs:decimal`-Argument und gibt einen `xs:dateTime`-Wert, der einen **zz**-Teil (Zeitzone) enthält, zurück. Das Ergebnis wird durch Berechnung des UTC **dateTime**-Äquivalents der Epochenzeit und Hinzufügen der (anhand der Systemuhr ermittelten) lokalen Zeitzone ermittelt. Wenn die Funktion z.B. auf einem Rechner, der in der Zeitzone +01:00 (relativ zur UTC) konfiguriert wurde, ausgeführt wird, so wird nach Berechnung des UTC-**dateTime**-Äquivalents eine Stunde zum Ergebnis

addiert. Auch die Zeitzoneninformation, die einen optionalen lexikalischen Bestandteil des **xs:dateTime**-Ergebnisses bildet, wird im **dateTime**-Ergebnis ausgegeben. Vergleichen Sie dieses Ergebnis mit dem von **dateTime-from-epoch-no-TZ** und auch der Funktion **epoch-from-dateTime**.

Beispiele

In den Beispielen unten wird eine lokale Zeitzone von UTC +01:00 angenommen. Das UTC **dateTime**-Äquivalent der angegebenen Epochenzeit wird folglich um eine Stunde erhöht. Die Zeitzone wird im Ergebnis ausgegeben.

- **altova:dateTime-from-epoch**(34) gibt 1970-01-01T01:00:34+01:00 zurück.
- **altova:dateTime-from-epoch**(62) gibt 1970-01-01T01:01:02+01:00 zurück.

dateTime-from-epoch-no-TZ [altova:]

altova:dateTime-from-epoch-no-TZ(Epoch *als xs:decimal* **als xs:dateTime** XP3.1 XQ3.1
Die Epochenzeit ist ein auf Unix-Systemen verwendetes Zeitsystem. Darin wird jeder Zeitpunkt als Anzahl der Sekunden seit 00:00:00 UTC des 1. Januar 1970 definiert. Die Funktion **dateTime-from-epoch-no-TZ** gibt das **xs:dateTime**-Äquivalent einer Epochenzeit zurück, passt es an die lokale Zeitzone an, inkludiert die Zeitzoneninformation jedoch nicht im Ergebnis.

Die Funktion erhält ein xs:decimal-Argument und gibt einen xs:dateTime-Wert, der keinen **zz**-Teil (Zeitzone) enthält, zurück. Das Ergebnis wird durch Berechnung des UTC **dateTime**-Äquivalents der Epochenzeit und Hinzufügen der (anhand der Systemuhr ermittelten) lokalen Zeitzone ermittelt. Wenn die Funktion z.B. auf einem Rechner, der in der Zeitzone +01:00 (relativ zur UTC) konfiguriert wurde, ausgeführt wird, so wird nach Berechnung des UTC-**dateTime**-Äquivalents eine Stunde zum Ergebnis addiert. Die Zeitzoneninformation, die einen optionalen lexikalischen Bestandteil des **xs:dateTime**-Ergebnisses bildet, wird nicht im **dateTime**-Ergebnis ausgegeben. Vergleichen Sie dieses Ergebnis mit dem von **dateTime-from-epoch** und auch der Funktion **epoch-from-dateTime**.

Beispiele

In den Beispielen unten wird eine lokale Zeitzone von UTC +01:00 angenommen. Das UTC **dateTime**-Äquivalent der angegebenen Epochenzeit wird folglich um eine Stunde erhöht. Die Zeitzone wird nicht im Ergebnis ausgegeben.

- **altova:dateTime-from-epoch**(34) gibt 1970-01-01T01:00:34 zurück.
- **altova:dateTime-from-epoch**(62) gibt 1970-01-01T01:01:02 zurück.

epoch-from-dateTime [altova:]

altova:epoch-from-dateTime(dateTimeValue *als xs:dateTime*) **als xs:decimal** XP3.1 XQ3.1
Die Epochenzeit ist ein auf Unix-Systemen verwendetes Zeitsystem. Darin wird jeder Zeitpunkt als Anzahl der Sekunden seit 00:00:00 UTC des 1. Januar 1970 definiert. Die Funktion **epoch-from-dateTime** gibt das Epochenzeitäquivalent von **xs:dateTime** zurück, welches als Argument der Funktion bereitgestellt wird. Beachten Sie, dass Sie den **xs:dateTime**-Wert eventuell explizit konstruieren müssen. Der angegebene **xs:dateTime**-Wert kann den optionalen **zz** (Zeitzonen)-Wert enthalten, muss ihn aber nicht enthalten.

Unabhängig davon, ob der Zeitzonenteil als Bestandteil des Arguments angegeben wird oder nicht, wird

der (anhand der Systemuhr ermittelte) lokale Zeitzonenunterschied vom angegebenen **dateTimeValue**-Argument subtrahiert. Dadurch wird das UTC-Zeit-Äquivalent erzeugt, anhand dessen die entsprechende Epochenzeit berechnet wird. Wenn die Funktion z.B. auf einem Rechner, der für die Zeitzone +01:00 (relativ zur UTC) konfiguriert wurde, ausgeführt wird, so wird vor Berechnung des Epochenzeitwerts eine Stunde vom angegebenen **dateTimeValue** subtrahiert. Siehe dazu auch die Funktion **dateTime-from-epoch**.

Beispiele

In den Beispielen unten wird eine lokale Zeitzone von UTC +01:00 angenommen. Daher wird vor Berechnung der Epochenzeit eine Stunde vom angegebenen **dateTime**-Wert subtrahiert.

- **altova:epoch-from-dateTime**(xs:dateTime("1970-01-01T01:00:34+01:00")) gibt 34 zurück.
- **altova:epoch-from-dateTime**(xs:dateTime("1970-01-01T01:00:34")) gibt 34 zurück.
- **altova:epoch-from-dateTime**(xs:dateTime("2021-04-01T11:22:33")) gibt 1617272553 zurück.

[Nach oben 439]

10.2.1.3 XPath/XQuery-Funktionen: Standort

Die folgenden XPath/XQuery-Erweiterungsfunktionen zu Standortdaten werden in der aktuellen Version von RaptorXML Server unterstützt und können in (i) in einem XSLT-Kontext in XPath-Ausdrücken oder (ii) in einem XQuery-Dokument in einem XQuery-Ausdruck verwendet werden.

Anmerkung zur Benennung von Funktionen und zur Anwendbarkeit der Sprache

Altova-Erweiterungsfunktionen können in XPath/XQuery-Ausdrücken verwendet werden. Dadurch stehen neben den Funktionen in der Standardbibliothek der XPath-, XQuery- und XSLT-Funktionen zusätzliche Funktionen zur Verfügung. Die Altova-Erweiterungsfunktionen befinden sich im **Altova-Erweiterungsfunktions-Namespace, http://www.altova.com/xslt-extensions** und sind in diesem Abschnitt mit dem Präfix **altova:**, das als an diesen Namespace gebunden angenommen wird, gekennzeichnet. Beachten Sie, dass manche Funktionen in zukünftigen Versionen Ihres Produkts eventuell nicht mehr unterstützt werden oder dass sich das Verhalten einzelner Funktionen ändern kann. Um zu sehen, welche Altova-Erweiterungsfunktionen unterstützt werden, lesen Sie bitte die Dokumentation zur jeweiligen Release.

XPath-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	XP1 XP2 XP3.1
XSLT-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	XSLT1 XSLT2 XSLT3
XQuery-Funktionen (in XQuery-Ausdrücken in XQuery verwendet):	XQ1 XQ3.1

format-geolocation [altova:]

altova:format-geolocation(Latitude *als xs:decimal*, **Longitude** *als xs:decimal*, **GeolocationOutputStringFormat** *als xs:integer*) als **xs:string** XP3.1 XQ3.1

Erhält als die ersten beiden Argumente die geografische Breite und Länge und gibt den Standort als String zurück. Das dritte Argument, **GeolocationOutputStringFormat**, ist das Format des Ausgabestring für den Standort; darin werden zum Identifizieren des Ausgabestringformats Ganzzahlwerte von 1 bis 4

verwendet (*siehe 'Format des Ausgabestrings für die geografische Position' weiter unten*). Die Werte für die Breite liegen im Bereich von `+90` bis `-90` (`N` nach `S`). Die Werte für die Länge liegen im Bereich von `+180` bis `-180` (`O` nach `W`).

Anmerkung: Zur Bereitstellung der Input-Strings können die Funktion `image-exif-data` [469] und die Attribute der Exif-Metadaten verwendet werden.

Beispiele

- **`altova:format-geolocation`**`(33.33, -22.22, 4)` gibt `xs:string "33.33 -22.22"` zurück
- **`altova:format-geolocation`**`(33.33, -22.22, 2)` gibt `xs:string "33.33N 22.22W"` zurück
- **`altova:format-geolocation`**`(-33.33, 22.22, 2)` gibt `xs:string "33.33S 22.22E"` zurück
- **`altova:format-geolocation`**`(33.33, -22.22, 1)` gibt `xs:string "33°19'48.00"S 22° 13'12.00"E"` zurück

Ausgabestringformate für die geografische Position:

Die bereitgestellte Breite und Länge ist in einem der unten aufgelisteten Ausgabeformate formatiert. Das gewünschte Format wird anhand seiner Ganzzahl-ID (`1` bis `4`) identifiziert. Die Breitenwerte liegen im Bereich von `+90` bis `-90` (`N` nach `S`). Die Längenwerte liegen im Bereich von `+180` bis `-180` (`O` nach `W`).

1

Grad, Minuten, Dezimalsekunden, mit nachgestellter Orientierung (`N/S, E/W`)
`D°M'S.SS"N/S D°M'S.SS"E/W`
Beispiel: **`33°55'11.11"N 22°44'66.66"W`**

2

Dezimalgrad, mit nachgestellter Orientierung (`N/S`, `E/W`)
`D.DDN/S D.DDE/W`
Beispiel: **`33.33N 22.22W`**

3

Grad, Minuten, Dezimalsekunden mit Vorzeichen (`+/-`); Plus-Zeichen für (**`N/E`**) ist optional
`+/-D°M'S.SS" +/-D°M'S.SS"`
Beispiel: **`33°55'11.11" -22°44'66.66"`**

4

Dezimalgrad, mit Vorzeichen (`+/-`); Plus-Zeichen für (**`N/E`**) ist optional
`+/-D.DD +/-D.DD`
Beispiel: **`33.33 -22.22`**

Altova Exif-Attribut: Geolocation

Der Altova XPath/XQuery-Prozessor generiert anhand der Exif-Standard-Metadaten-Tags das benutzerdefinierte Attribut **`Geolocation`**. **`Geolocation`** ist eine Verkettung von vier Exif-Tags: `GPSLatitude`, `GPSLatitudeRef`, `GPSLongitude`, `GPSLongitudeRef` mit hinzugefügten Werten (*siehe*

Tabelle unten).

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

parse-geolocation [altova:]

altova:parse-geolocation(GeolocationInputString *als xs:string*) als **xs:decimal+** **XP3.1** **XQ3.1**
Parst das bereitgestellte `GeolocationInputString`-Argument und gibt die geografische Breite und Länge (in dieser Reihenfolge) als Sequenz aus zwei `xs:decimal` Elementen zurück. Die Formate, in denen der Input-String für die geografische Position bereitgestellt werden kann, sind unten aufgelistet.

Anmerkung: Zur Bereitstellung des Input-String für die geografische Position können die Funktion image-exif-data[469] und das @Geolocation[469]-Attribut der Exif-Metadaten verwendet werden (*siehe Beispiel unten*).

Beispiele

- **altova:parse-geolocation**("33.33 -22.22") gibt die Sequenz bestehend aus zwei `xs:decimals (33.33, 22.22)` Elementen zurück
- **altova:parse-geolocation**("48°51'29.6""N 24°17'40.2""") gibt die Sequenz bestehend aus zwei `xs:decimals (48.8582222222222, 24.2945)` Elementen zurück
- **altova:parse-geolocation**('48°51''29.6"N 24°17''40.2"') gibt die Sequenz bestehend aus zwei `xs:decimals (48.8582222222222, 24.2945)` Elementen zurück
- **altova:parse-geolocation**(**image-exif-data**(//MyImages/Image20141130.01)**/@Geolocation**) gibt die Sequenz bestehend aus zwei `xs:decimals` Elementen zurück

Input-String-Formate der Standortdaten:

Der Input-String für die geografische Position muss die Breite und Länge (in dieser Reihenfolge) getrennt durch ein Leerzeichen enthalten. Beide Werte können jedes der folgenden Formate haben. Auch Kombinationen sind zulässig, d.h. die Breite kann in einem anderen Format als die Länge angegeben werden. Die Breitenwerte liegen im Bereich `+90` bis `-90` (`N` nach `S`). Die Längenwerte liegen im Bereich von `+180` bis `-180` (`O` nach `W`).

Anmerkung: Wenn als Trennzeichen für den Input-String einfache oder doppelte Anführungszeichen verwendet werden, kann dies zu einer Fehlinterpretation der einfachen bzw. doppelten Anführungszeichen als Minuten- bzw. Sekundenwerte führen. In solchen Fällen müssen die zur Angabe der Minuten- und Sekundenwerte verwendeten Anführungszeichen durch Verdoppelung mit einem Escape-Zeichen versehen werden. In den Beispielen in diesem Abschnitt sind Anführungszeichen, die als Trennzeichen für den Input-String dienen, gelb markiert ("), während Maßeinheitsangaben blau ("") markiert sind.

- Grad, Minuten, Dezimalsekunden, mit nachgestellter Orientierung (`N/S, O/W`)
  D°M'S.SS"N/S D°M'S.SS"W/E

Beispiel: `33°55'11.11"N 22°44'55.25"W`

- Grad, Minuten, Dezimalsekunden mit Vorzeichen (`+/-`); das Plus-Zeichen für (`N`/`O`) ist optional
  `+/-D°M'S.SS" +/-D°M'S.SS"`
  Beispiel: `33°55'11.11" -22°44'55.25"`

- Grad, Dezimalminuten mit nachgestellter Orientierung (`N/S`, `O/W`)
  `D°M.MM'N/S D°M.MM'W/E`
  Beispiel: `33°55.55'N 22°44.44'W`

- Grad, Dezimalminuten mit Vorzeichen (`+/-`); das Plus-Zeichen für (`N`/`O`) ist optional
  `+/-D°M.MM' +/-D°M.MM'`
  Beispiel: `+33°55.55' -22°44.44'`

- Dezimalgrade, mit nachgestellter Orientierung (`N/S`, `O/W`)
  `D.DDN/S D.DDW/E`
  Beispiel: `33.33N 22.22W`

- Dezimalgrade mit Vorzeichen (`+/-`); das Plus-Zeichen für (`N/S` `O/W`) ist optional
  `+/-D.DD +/-D.DD`
  Beispiel: `33.33 -22.22`

Beispiele für Formatkombinationen:

```
33.33N -22°44'55.25"
33.33 22°44'55.25"W
33.33 22.45
```

Altova Exif-Attribut: Geolocation

Der Altova XPath/XQuery-Prozessor generiert anhand der Exif-Standard-Metadaten-Tags das benutzerdefinierte Attribut `Geolocation`. `Geolocation` ist eine Verkettung von vier Exif-Tags: `GPSLatitude`, `GPSLatitudeRef`, `GPSLongitude`, `GPSLongitudeRef` mit hinzugefügten Werten (*siehe Tabelle unten*).

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

geolocation-distance-km [altova:]

altova:geolocation-distance-km(GeolocationInputString-1 *als xs:string*, **GeolocationInputString-2** *als xs:string*) als **xs:decimal** **XP3.1** **XQ3.1**

Berechnet die Entfernung zwischen zwei geografischen Positionen in Kilometern. Die Formate, in denen der Input-String für die geografischen Position angegeben werden kann, sind unten aufgelistet. Die Breitenwerte liegen im Bereich von `+90` bis `-90` (`N` nach `S`). Die Längenwerte liegen im Bereich von `+180` bis `-180` (`O` nach `W`).

Anmerkung: Zur Bereitstellung des Input-String für die geografische Position können die Funktion image-exif-data[469] und das @Geolocation[469]-Attribut der Exif-Metadaten verwendet werden.

Beispiele

- `altova:geolocation-distance-km("33.33 -22.22", "48°51'29.6""N 24°17'40.2""")` gibt `xs:decimal 4183.08132372392` zurück

Input-String-Formate der Standortdaten:

Der Input-String für die geografische Position muss die Breite und Länge (in dieser Reihenfolge) getrennt durch ein Leerzeichen enthalten. Beide Werte können jedes der folgenden Formate haben. Auch Kombinationen sind zulässig, d.h. die Breite kann in einem anderen Format als die Länge angegeben werden. Die Breitenwerte liegen im Bereich `+90` bis `-90` (`N` nach `S`). Die Längenwerte liegen im Bereich von `+180` bis `-180` (`O` nach `W`).

Anmerkung: Wenn als Trennzeichen für den Input-String einfache oder doppelte Anführungszeichen verwendet werden, kann dies zu einer Fehlinterpretation der einfachen bzw. doppelten Anführungszeichen als Minuten- bzw. Sekundenwerte führen. In solchen Fällen müssen die zur Angabe der Minuten- und Sekundenwerte verwendeten Anführungszeichen durch Verdoppelung mit einem Escape-Zeichen versehen werden. In den Beispielen in diesem Abschnitt sind Anführungszeichen, die als Trennzeichen für den Input-String dienen, gelb markiert (`"`), während Maßeinheitsangaben blau (`""`) markiert sind.

- Grad, Minuten, Dezimalsekunden, mit nachgestellter Orientierung (`N/S, O/W`)
  `D°M'S.SS"N/S D°M'S.SS"W/E`
  Beispiel: **`33°55'11.11"N 22°44'55.25"W`**

- Grad, Minuten, Dezimalsekunden mit Vorzeichen (`+/-`); das Plus-Zeichen für (**`N/O`**) ist optional
  `+/-D°M'S.SS" +/-D°M'S.SS"`
  Beispiel: **`33°55'11.11" -22°44'55.25"`**

- Grad, Dezimalminuten mit nachgestellter Orientierung (`N/S`, `O/W`)
  `D°M.MM'N/S D°M.MM'W/E`
  Beispiel: **`33°55.55'N 22°44.44'W`**

- Grad, Dezimalminuten mit Vorzeichen (`+/-`); das Plus-Zeichen für (**`N/O`**) ist optional
  `+/-D°M.MM' +/-D°M.MM'`
  Beispiel: **`+33°55.55' -22°44.44'`**

- Dezimalgrade, mit nachgestellter Orientierung (`N/S`, `O/W`)
  `D.DDN/S D.DDW/E`
  Beispiel: **`33.33N 22.22W`**

- Dezimalgrade mit Vorzeichen (`+/-`); das Plus-Zeichen für (**`N/S O/W`**) ist optional
  `+/-D.DD +/-D.DD`
  Beispiel: **`33.33 -22.22`**

Beispiele für Formatkombinationen:

`33.33N -22°44'55.25"`
`33.33 22°44'55.25"W`

```
33.33  22.45
```

Altova Exif-Attribut: Geolocation

Der Altova XPath/XQuery-Prozessor generiert anhand der Exif-Standard-Metadaten-Tags das benutzerdefinierte Attribut `Geolocation`. `Geolocation` ist eine Verkettung von vier Exif-Tags: `GPSLatitude`, `GPSLatitudeRef`, `GPSLongitude`, `GPSLongitudeRef` mit hinzugefügten Werten (*siehe Tabelle unten*).

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

geolocation-distance-mi [altova:]

altova:geolocation-distance-mi(GeolocationInputString-1 *als xs:string*, **GeolocationInputString-2** *als xs:string*) als **xs:decimal** **XP3.1** **XQ3.1**

Berechnet die Entfernung zwischen zwei geografischen Positionen in Meilen. Die Formate, in denen der Input-String für die geografischen Position angegeben werden kann, sind unten aufgelistet. Die Breitenwerte liegen im Bereich von `+90` bis `-90` (`N` nach `S`). Die Längenwerte liegen im Bereich von `+180` bis `-180` (`O` nach `W`).

Anmerkung: Zur Bereitstellung des Input-String für die geografische Position können die Funktion `image-exif-data` [469] und das `@Geolocation` [469]-Attribut der Exif-Metadaten verwendet werden.

Beispiele

- `altova:geolocation-distance-mi("33.33  -22.22", "48°51'29.6""N  24°17'40.2""")` gibt `xs:decimal 2599.40652340653` zurück

Input-String-Formate der Standortdaten:

Der Input-String für die geografische Position muss die Breite und Länge (in dieser Reihenfolge) getrennt durch ein Leerzeichen enthalten. Beide Werte können jedes der folgenden Formate haben. Auch Kombinationen sind zulässig, d.h. die Breite kann in einem anderen Format als die Länge angegeben werden. Die Breitenwerte liegen im Bereich `+90` bis `-90` (`N` nach `S`). Die Längenwerte liegen im Bereich von `+180` bis `-180` (`O` nach `W`).

Anmerkung: Wenn als Trennzeichen für den Input-String einfache oder doppelte Anführungszeichen verwendet werden, kann dies zu einer Fehlinterpretation der einfachen bzw. doppelten Anführungszeichen als Minuten- bzw. Sekundenwerte führen. In solchen Fällen müssen die zur Angabe der Minuten- und Sekundenwerte verwendeten Anführungszeichen durch Verdoppelung mit einem Escape-Zeichen versehen werden. In den Beispielen in diesem Abschnitt sind Anführungszeichen, die als Trennzeichen für den Input-String dienen, gelb markiert (`"`), während Maßeinheitsangaben blau (`""`) markiert sind.

- Grad, Minuten, Dezimalsekunden, mit nachgestellter Orientierung (`N/S, O/W`)
  `D°M'S.SS"N/S  D°M'S.SS"W/E`

Beispiel: `33°55'11.11"N 22°44'55.25"W`

- Grad, Minuten, Dezimalsekunden mit Vorzeichen (+/-); das Plus-Zeichen für (N/O) ist optional
  `+/-D°M'S.SS" +/-D°M'S.SS"`
  Beispiel: `33°55'11.11" -22°44'55.25"`

- Grad, Dezimalminuten mit nachgestellter Orientierung (N/S, O/W)
  `D°M.MM'N/S D°M.MM'W/E`
  Beispiel: `33°55.55'N 22°44.44'W`

- Grad, Dezimalminuten mit Vorzeichen (+/-); das Plus-Zeichen für (N/O) ist optional
  `+/-D°M.MM' +/-D°M.MM'`
  Beispiel: `+33°55.55' -22°44.44'`

- Dezimalgrade, mit nachgestellter Orientierung (N/S, O/W)
  `D.DDN/S D.DDW/E`
  Beispiel: `33.33N 22.22W`

- Dezimalgrade mit Vorzeichen (+/-); das Plus-Zeichen für (N/S O/W) ist optional
  `+/-D.DD +/-D.DD`
  Beispiel: `33.33 -22.22`

Beispiele für Formatkombinationen:

```
33.33N -22°44'55.25"
33.33 22°44'55.25"W
33.33 22.45
```

Altova Exif-Attribut: Geolocation

Der Altova XPath/XQuery-Prozessor generiert anhand der Exif-Standard-Metadaten-Tags das benutzerdefinierte Attribut `Geolocation`. `Geolocation` ist eine Verkettung von vier Exif-Tags: `GPSLatitude`, `GPSLatitudeRef`, `GPSLongitude`, `GPSLongitudeRef` mit hinzugefügten Werten (*siehe Tabelle unten*).

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

geolocations-bounding-rectangle [altova:]

altova:geolocations-bounding-rectangle(Geolocations *als xs:sequence*, **GeolocationOutputStringFormat** *als xs:integer*) als **xs:string** **XP3.1** **XQ3.1**
Erhält als erstes Argument eine Sequenz von Strings, wobei es sich bei jedem String in der Sequenz um eine geografische Position handelt. Die Funktion gibt eine Sequenz von zwei Strings zurück, die die geografischen Positionskoordinaten der linken oberen bzw. rechten unteren Ecke eines Rechtecks bilden, dessen Größe so angepasst ist, dass es alle im ersten Argument angegebenen Positionskoordinaten

enthält. Die Formate, in denen der Input-String für die geografischen Position angegeben werden kann, sind unten aufgelistet (*siehe 'Input-String-Formate der Standortdaten'*). Die Breitenwerte liegen im Bereich von `+90` bis `-90` (`N` nach `S`). Die Längenwerte liegen im Bereich von `+180` bis `-180` (`O` nach `W`).

Im zweiten Argument der Funktion ist das Format der beiden Geolocation-Strings in der Ausgabesequenz angegeben. Das Argument erhält einen Ganzzahlwert von `1` bis `4`, wobei die einzelnen Werte ein jeweils unterschiedliches String-Format definieren (*siehe 'Ausgabestringsformate für die geografische Position' weiter unten).*

Anmerkung: Zur Bereitstellung der Input-Strings können die Funktion `image-exif-data` [469] und die Attribute der Exif-Metadaten verwendet werden.

Beispiele

- **`altova:geolocations-bounding-rectangle`**`(("48.2143531 16.3707266", "51.50939 -0.11832"), 1)` gibt die Sequenz `("51°30'33.804"N 0°7'5.952"W", "48°12'51.67116"N 16°22'14.61576"E")` zurück.
- **`altova:geolocations-bounding-rectangle`**`(("48.2143531 16.3707266", "51.50939 -0.11832", "42.5584577 -70.8893334"), 4)` gibt die Sequenz `("51.50939 -70.8893334", "42.5584577 16.3707266")` zurück.

Input-String-Formate der Standortdaten:

Der Input-String für die geografische Position muss die Breite und Länge (in dieser Reihenfolge) getrennt durch ein Leerzeichen enthalten. Beide Werte können jedes der folgenden Formate haben. Auch Kombinationen sind zulässig, d.h. die Breite kann in einem anderen Format als die Länge angegeben werden. Die Breitenwerte liegen im Bereich `+90` bis `-90` (`N` nach `S`). Die Längenwerte liegen im Bereich von `+180` bis `-180` (`O` nach `W`).

Anmerkung: Wenn als Trennzeichen für den Input-String einfache oder doppelte Anführungszeichen verwendet werden, kann dies zu einer Fehlinterpretation der einfachen bzw. doppelten Anführungszeichen als Minuten- bzw. Sekundenwerte führen. In solchen Fällen müssen die zur Angabe der Minuten- und Sekundenwerte verwendeten Anführungszeichen durch Verdoppelung mit einem Escape-Zeichen versehen werden. In den Beispielen in diesem Abschnitt sind Anführungszeichen, die als Trennzeichen für den Input-String dienen, gelb markiert (`"`), während Maßeinheitsangaben blau (`""`) markiert sind.

- Grad, Minuten, Dezimalsekunden, mit nachgestellter Orientierung (`N/S, O/W`)
  `D°M'S.SS"N/S D°M'S.SS"W/E`
  Beispiel: **`33°55'11.11"N 22°44'55.25"W`**

- Grad, Minuten, Dezimalsekunden mit Vorzeichen (`+/-`); das Plus-Zeichen für (**`N`**/**`O`**) ist optional
  `+/-D°M'S.SS" +/-D°M'S.SS"`
  Beispiel: **`33°55'11.11" -22°44'55.25"`**

- Grad, Dezimalminuten mit nachgestellter Orientierung (`N/S`, `O/W`)
  `D°M.MM'N/S D°M.MM'W/E`
  Beispiel: **`33°55.55'N 22°44.44'W`**

- Grad, Dezimalminuten mit Vorzeichen (`+/-`); das Plus-Zeichen für (**`N`**/**`O`**) ist optional
  `+/-D°M.MM' +/-D°M.MM'`

Beispiel: `+33°55.55' -22°44.44'`

- Dezimalgrade, mit nachgestellter Orientierung (`N/S`, `O/W`)
  `D.DDN/S D.DDW/E`
  Beispiel: `33.33N 22.22W`

- Dezimalgrade mit Vorzeichen (`+/-`); das Plus-Zeichen für (`N/S O/W`) ist optional
  `+/-D.DD +/-D.DD`
  Beispiel: `33.33 -22.22`

Beispiele für Formatkombinationen:

```
33.33N -22°44'55.25"
33.33 22°44'55.25"W
33.33 22.45
```

Ausgabestringformate für die geografische Position:

Die bereitgestellte Breite und Länge ist in einem der unten aufgelisteten Ausgabeformate formatiert. Das gewünschte Format wird anhand seiner Ganzzahl-ID (`1` bis `4`) identifiziert. Die Breitenwerte liegen im Bereich von `+90` bis `-90` (`N` nach `S`). Die Längenwerte liegen im Bereich von `+180` bis `-180` (`O` nach `W`).

1
Grad, Minuten, Dezimalsekunden, mit nachgestellter Orientierung (`N/S, E/W`) `D°M'S.SS"N/S D°M'S.SS"E/W` *Beispiel*: `33°55'11.11"N 22°44'66.66"W`

2
Dezimalgrad, mit nachgestellter Orientierung (`N/S`, `E/W`) `D.DDN/S D.DDE/W` *Beispiel*: `33.33N 22.22W`

3
Grad, Minuten, Dezimalsekunden mit Vorzeichen (`+/-`); Plus-Zeichen für (`N/E`) ist optional `+/-D°M'S.SS" +/-D°M'S.SS"` *Beispiel*: `33°55'11.11" -22°44'66.66"`

4
Dezimalgrad, mit Vorzeichen (`+/-`); Plus-Zeichen für (`N/E`) ist optional `+/-D.DD +/-D.DD` *Beispiel*: `33.33 -22.22`

Altova Exif-Attribut: Geolocation

Der Altova XPath/XQuery-Prozessor generiert anhand der Exif-Standard-Metadaten-Tags das benutzerdefinierte Attribut `Geolocation`. `Geolocation` ist eine Verkettung von vier Exif-Tags: `GPSLatitude`, `GPSLatitudeRef`, `GPSLongitude`, `GPSLongitudeRef` mit hinzugefügten Werten (*siehe*

Tabelle unten).

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

▼ geolocation-within-polygon [altova:]

altova:geolocation-within-polygon(Geolocation *als xs:string*, **((PolygonPoint** *als xs:string*)+)) als **xs:boolean** **XP3.1** **XQ3.1**

Ermittelt ob sich **Geolocation** (das erste Argument) innerhalb des durch die **PolygonPoint**-Argumente beschriebenen Polygonbereichs befindet. Wenn die PolygonPoint-Argumente keine geschlossene Form (wenn der erste und der letzte Punkt identisch sind) bilden, so wird der erste Punkt implizit zum letzten Punkt hinzugefügt, um die Form zu schließen. Alle Argumente (Geolocation und PolygonPoint+) werden durch Input-Strings für die geografische Position (*Formatliste siehe unten*) angegeben. Wenn sich das Geolocation Argument innerhalb des Polygons befindet, gibt die Funktion true() zurück; andernfalls gibt sie false() zurück. Die Breitenwerte liegen im Bereich von +90 bis -90 (N nach S). Die Längenwerte liegen im Bereich von +180 bis -180 (O nach W).

Anmerkung: Zur Bereitstellung des Input-String für die geografische Position können die Funktion image-exif-data[469] und das @Geolocation[469]-Attribut der Exif-Metadaten verwendet werden.

⊟ *Beispiele*

- **altova:geolocation-within-polygon("33 -22", ("58 -32", "-78 -55", "48 24", "58 -32"))** gibt true() zurück
- **altova:geolocation-within-polygon("33 -22", ("58 -32", "-78 -55", "48 24"))** gibt true() zurück
- **altova:geolocation-within-polygon("33 -22", ("58 -32", "-78 -55", "48°51'29.6""N 24°17'40.2""))** gibt true() zurück

⊟ *Input-String-Formate der Standortdaten:*

Der Input-String für die geografische Position muss die Breite und Länge (in dieser Reihenfolge) getrennt durch ein Leerzeichen enthalten. Beide Werte können jedes der folgenden Formate haben. Auch Kombinationen sind zulässig, d.h. die Breite kann in einem anderen Format als die Länge angegeben werden. Die Breitenwerte liegen im Bereich +90 bis -90 (N nach S). Die Längenwerte liegen im Bereich von +180 bis -180 (O nach W).

Anmerkung: Wenn als Trennzeichen für den Input-String einfache oder doppelte Anführungszeichen verwendet werden, kann dies zu einer Fehlinterpretation der einfachen bzw. doppelten Anführungszeichen als Minuten- bzw. Sekundenwerte führen. In solchen Fällen müssen die zur Angabe der Minuten- und Sekundenwerte verwendeten Anführungszeichen durch Verdoppelung mit einem Escape-Zeichen versehen werden. In den Beispielen in diesem Abschnitt sind Anführungszeichen, die als Trennzeichen für den Input-String dienen, gelb markiert ("), während Maßeinheitsangaben blau ("") markiert sind.

- Grad, Minuten, Dezimalsekunden, mit nachgestellter Orientierung (N/S, O/W)
  D°M'S.SS"N/S D°M'S.SS"W/E
  Beispiel: **33°55'11.11"N 22°44'55.25"W**

- Grad, Minuten, Dezimalsekunden mit Vorzeichen (+/-); das Plus-Zeichen für (**N**/O) ist optional
  +/-D°M'S.SS" +/-D°M'S.SS"
  Beispiel: **33°55'11.11" -22°44'55.25"**

- Grad, Dezimalminuten mit nachgestellter Orientierung (N/S, O/W)
  D°M.MM'N/S D°M.MM'W/E
  Beispiel: **33°55.55'N 22°44.44'W**

- Grad, Dezimalminuten mit Vorzeichen (+/-); das Plus-Zeichen für (**N**/O) ist optional
  +/-D°M.MM' +/-D°M.MM'
  Beispiel: **+33°55.55' -22°44.44'**

- Dezimalgrade, mit nachgestellter Orientierung (N/S, O/W)
  D.DDN/S D.DDW/E
  Beispiel: **33.33N 22.22W**

- Dezimalgrade mit Vorzeichen (+/-); das Plus-Zeichen für (**N/S O/W**) ist optional
  +/-D.DD +/-D.DD
  Beispiel: **33.33 -22.22**

Beispiele für Formatkombinationen:
33.33N -22°44'55.25"
33.33 22°44'55.25"W
33.33 22.45

- *Altova Exif-Attribut: Geolocation*

  Der Altova XPath/XQuery-Prozessor generiert anhand der Exif-Standard-Metadaten-Tags das benutzerdefinierte Attribut **Geolocation**. **Geolocation** ist eine Verkettung von vier Exif-Tags: GPSLatitude, GPSLatitudeRef, GPSLongitude, GPSLongitudeRef mit hinzugefügten Werten (*siehe Tabelle unten*).

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

- geolocation-within-rectangle [altova:]

  altova:geolocation-within-rectangle(Geolocation *als xs:string*, **RectCorner-1** *als xs:string*, **RectCorner-2** *als xs:string*) als **xs:boolean** **XP3.1** **XQ3.1**
  Ermittelt, ob sich **Geolocation** (das erste Argument) innerhalb des durch das zweite und dritte Argument, **RectCorner-1** und **RectCorner-2**, definierten Rechtecks befindet. **RectCorner-1** und **RectCorner-2**

definieren gegenüberliegende Eckpunkte des Rechtecks. Alle Argumente (`Geolocation`, **`RectCorner-1`** und **`RectCorner-2`**) werden durch Input-Strings für die geografische Position (*Formatliste siehe unten*) angegeben. Wenn sich das `Geolocation`-Argument innerhalb des Rechtecks befindet, gibt die Funktion `true()` zurück; andernfalls gibt sie `false()` zurück. Die Breitenwerte liegen im Bereich von `+90` bis `-90` (`N` nach `S`). Die Längenwerte liegen im Bereich von `+180` bis `-180` (`O` nach `W`).

Anmerkung: Zur Bereitstellung des Input-String für die geografische Position können die Funktion `image-exif-data` 469 und das `@Geolocation` 469-Attribut der Exif-Metadaten verwendet werden.

Beispiele

- **`altova:geolocation-within-rectangle(`**`"33 -22", "58 -32", "-48 24"`**`)`** gibt `true()` zurück
- **`altova:geolocation-within-rectangle(`**`"33 -22", "58 -32", "48 24"`**`)`** gibt `false()` zurück
- **`altova:geolocation-within-rectangle(`**`"33 -22", "58 -32", "48°51'29.6""S 24° 17'40.2""`**`)`** gibt `true()` zurück

Input-String-Formate der Standortdaten:

Der Input-String für die geografische Position muss die Breite und Länge (in dieser Reihenfolge) getrennt durch ein Leerzeichen enthalten. Beide Werte können jedes der folgenden Formate haben. Auch Kombinationen sind zulässig, d.h. die Breite kann in einem anderen Format als die Länge angegeben werden. Die Breitenwerte liegen im Bereich `+90` bis `-90` (`N` nach `S`). Die Längenwerte liegen im Bereich von `+180` bis `-180` (`O` nach `W`).

Anmerkung: Wenn als Trennzeichen für den Input-String einfache oder doppelte Anführungszeichen verwendet werden, kann dies zu einer Fehlinterpretation der einfachen bzw. doppelten Anführungszeichen als Minuten- bzw. Sekundenwerte führen. In solchen Fällen müssen die zur Angabe der Minuten- und Sekundenwerte verwendeten Anführungszeichen durch Verdoppelung mit einem Escape-Zeichen versehen werden. In den Beispielen in diesem Abschnitt sind Anführungszeichen, die als Trennzeichen für den Input-String dienen, gelb markiert (`"`), während Maßeinheitsangaben blau (`""`) markiert sind.

- Grad, Minuten, Dezimalsekunden, mit nachgestellter Orientierung (`N/S, O/W`)
  `D°M'S.SS"N/S D°M'S.SS"W/E`
  Beispiel: **`33°55'11.11"N 22°44'55.25"W`**

- Grad, Minuten, Dezimalsekunden mit Vorzeichen (`+/-`); das Plus-Zeichen für (**`N`**/`O`) ist optional
  `+/-D°M'S.SS" +/-D°M'S.SS"`
  Beispiel: **`33°55'11.11" -22°44'55.25"`**

- Grad, Dezimalminuten mit nachgestellter Orientierung (`N/S`, `O/W`)
  `D°M.MM'N/S D°M.MM'W/E`
  Beispiel: **`33°55.55'N 22°44.44'W`**

- Grad, Dezimalminuten mit Vorzeichen (`+/-`); das Plus-Zeichen für (**`N`**/`O`) ist optional
  `+/-D°M.MM' +/-D°M.MM'`
  Beispiel: **`+33°55.55' -22°44.44'`**

- Dezimalgrade, mit nachgestellter Orientierung (`N/S`, `O/W`)
  `D.DDN/S D.DDW/E`

Beispiel: `33.33N 22.22W`

- Dezimalgrade mit Vorzeichen (`+/-`); das Plus-Zeichen für (`N/S o/W`) ist optional
  `+/-D.DD +/-D.DD`
  Beispiel: `33.33 -22.22`

Beispiele für Formatkombinationen:

```
33.33N -22°44'55.25"
33.33 22°44'55.25"W
33.33 22.45
```

- *Altova Exif-Attribut: Geolocation*

  Der Altova XPath/XQuery-Prozessor generiert anhand der Exif-Standard-Metadaten-Tags das benutzerdefinierte Attribut `Geolocation`. `Geolocation` ist eine Verkettung von vier Exif-Tags: `GPSLatitude`, `GPSLatitudeRef`, `GPSLongitude`, `GPSLongitudeRef` mit hinzugefügten Werten (*siehe Tabelle unten*).

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

[Nach oben 457]

10.2.1.4 XPath/XQuery-Funktionen: Bildbezogene

Die folgenden XPath/XQuery-Erweiterungsfunktionen im Zusammenhang mit Bildern werden in der aktuellen Version von RaptorXML Server unterstützt und können in (i) in einem XSLT-Kontext in XPath-Ausdrücken oder (ii) in einem XQuery-Dokument in einem XQuery-Ausdruck verwendet werden.

Anmerkung zur Benennung von Funktionen und zur Anwendbarkeit der Sprache

Altova-Erweiterungsfunktionen können in XPath/XQuery-Ausdrücken verwendet werden. Dadurch stehen neben den Funktionen in der Standardbibliothek der XPath-, XQuery- und XSLT-Funktionen zusätzliche Funktionen zur Verfügung. Die Altova-Erweiterungsfunktionen befinden sich im **Altova-Erweiterungsfunktions-Namespace,** `http://www.altova.com/xslt-extensions` und sind in diesem Abschnitt mit dem Präfix `altova:`, das als an diesen Namespace gebunden angenommen wird, gekennzeichnet. Beachten Sie, dass manche Funktionen in zukünftigen Versionen Ihres Produkts eventuell nicht mehr unterstützt werden oder dass sich das Verhalten einzelner Funktionen ändern kann. Um zu sehen, welche Altova-Erweiterungsfunktionen unterstützt werden, lesen Sie bitte die Dokumentation zur jeweiligen Release.

XPath-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	XP1 XP2 XP3.1

XSLT-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	XSLT1 XSLT2 XSLT3
XQuery-Funktionen (in XQuery-Ausdrücken in XQuery verwendet):	XQ1 XQ3.1

suggested-image-file-extension [altova:]

altova:suggested-image-file-extension(Base64String *als string*) als **string?** XP3.1 XQ3.1
Erhält die Base64-Kodierung einer Bilddatei als Argument und gibt die darin enthaltene Dateierweiterung des Bilds zurück. Der Rückgabewert ist ein Vorschlag, basierend auf den in der Kodierung enthaltenen Bilddateitypinformationen. Wenn diese Informationen nicht verfügbar sind, wird ein leerer String zurückgegeben. Diese Funktion ist nützlich, wenn Sie ein Base64-Bild als Datei speichern und die entsprechende Dateierweiterung dynamisch abrufen möchten.

Beispiele

- `altova:suggested-image-file-extension(/MyImages/MobilePhone/Image20141130.01)` gibt `'jpg'` zurück
- `altova:suggested-image-file-extension($XML1/Staff/Person/@photo)` gibt `''` zurück

In den Beispielen oben wird von den als Argument der Funktion bereitgestellten Nodes angenommen, dass sie ein Base64-kodiertes Bild enthalten. Im ersten Beispiel wird `jpg` als Dateityp bzw. Dateierweiterung abgerufen. Im zweiten Beispiel enthält die angegebene Base54-Kodierung keine brauchbaren Dateierweiterungsinformationen.

image-exif-data [altova:]

altova:image-exif-data(Base64BinaryString *als string*) als **element?** XP3.1 XQ3.1
Erhält ein Base64-kodiertes JPEG-Bild als Argument und gibt ein Element namens **Exif** zurück, das die Exif-Metadaten des Bilds enthält. Die Exif-Metadaten werden als Attribut-Wert-Paare des `Exif`-Elements erstellt. Bei den Attributnamen handelt es sich um die Exif-Daten-Tags aus der Base64-Kodierung. Weiter unten sehen Sie eine Liste der Exif-Tags. Wenn die Exif-Daten einen anbieterspezifischen Tag enthalten, so wird auch dieser Tag und sein Wert als Attribut-Wert-Paar zurückgegeben. Zusätzlich zu den Standard-Exif-Metadatentags (*siehe Liste unten*) werden auch Altova-spezifische Attribut-Wert-Paare generiert. Diese Altova Exif-Attribute sind unten aufgelistet.

Beispiele

- Um ein einziges Attribut abzurufen, verwenden Sie die Funktion folgendermaßen:
  `image-exif-data(//MyImages/Image20141130.01)/@GPSLatitude`
  `image-exif-data(//MyImages/Image20141130.01)/@Geolocation`
- Um alle Attribute abzurufen, verwenden Sie die Funktion folgendermaßen:
  `image-exif-data(//MyImages/Image20141130.01)/@*`
- Um die Namen aller Attribute abzurufen, verwenden Sie den folgenden Ausdruck:
  `for $i in image-exif-data(//MyImages/Image20141130.01)/@* return name($i)`
  Auf diese Art können Sie die Namen der von der Funktion zurückgegebenen Attribute eruieren.

Altova Exif-Attribut: Geolocation

Der Altova XPath/XQuery-Prozessor generiert anhand der Exif-Standard-Metadaten-Tags das benutzerdefinierte Attribut **Geolocation**. **Geolocation** ist eine Verkettung von vier Exif-Tags:

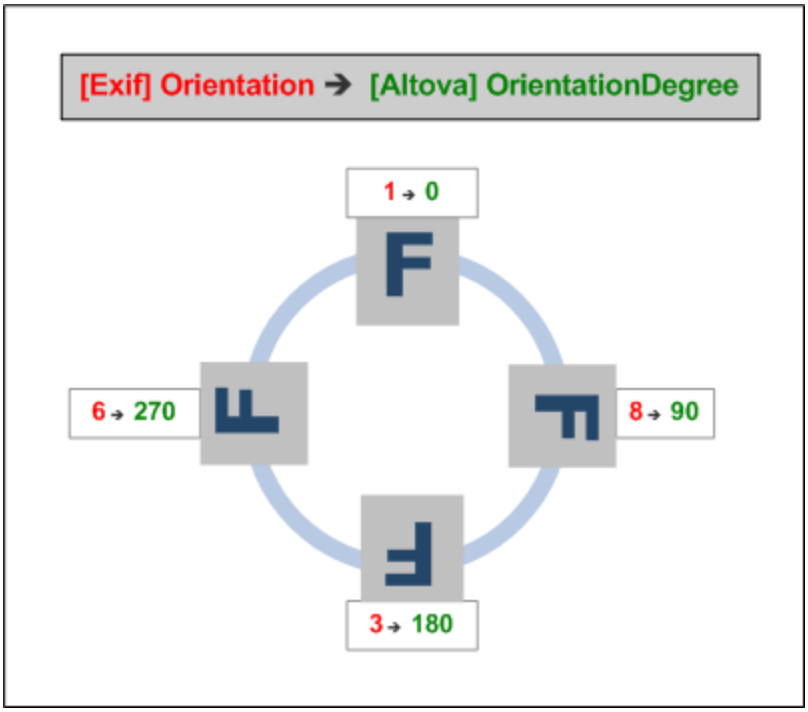
`GPSLatitude`, `GPSLatitudeRef`, `GPSLongitude`, `GPSLongitudeRef` mit hinzugefügten Werten (*siehe Tabelle unten*).

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

- *Altova Exif-Attribut: OrientationDegree*

Der Altova XPath/XQuery-Prozessor generiert anhand des Exif-Metadaten-Tags **`Orientation`** das benutzerdefinierte Attribut **`OrientationDegree`**.

`OrientationDegree` übersetzt den Standard-Exif-Tag `Orientation` von einem Ganzzahlwert (`1`, `8`, `3` oder `6`) in die entsprechenden Gradwerte dafür (`0`, `90`, `180`, `270`) (*siehe Abbildung unten*). Beachten Sie dass es keine Übersetzung der `Orientation`-Werte `2`, `4`, `5`, `7` gibt. (Diese Ausrichtungen werden durch Spiegelung des Bilds `1` an seiner senkrechten Mittelachse zur Erzeugung des Bilds mit dem Wert `2` und anschließende Drehung dieses Bilds um jeweils 90 Grad zur Erzeugung der Werte `7` bzw. `4` bzw. `5` erzielt).



- *Liste der Standard-Exif-Metatags*

- `ImageWidth`
- `ImageLength`
- `BitsPerSample`

- Compression
- PhotometricInterpretation
- Orientation
- SamplesPerPixel
- PlanarConfiguration
- YCbCrSubSampling
- YCbCrPositioning
- XResolution
- YResolution
- ResolutionUnit
- StripOffsets
- RowsPerStrip
- StripByteCounts
- JPEGInterchangeFormat
- JPEGInterchangeFormatLength
- TransferFunction
- WhitePoint
- PrimaryChromaticities
- YCbCrCoefficients
- ReferenceBlackWhite
- DateTime
- ImageDescription
- Make
- Model
- Software
- Artist
- Copyright

- ExifVersion
- FlashpixVersion
- ColorSpace
- ComponentsConfiguration
- CompressedBitsPerPixel
- PixelXDimension
- PixelYDimension
- MakerNote
- UserComment
- RelatedSoundFile
- DateTimeOriginal
- DateTimeDigitized
- SubSecTime
- SubSecTimeOriginal
- SubSecTimeDigitized
- ExposureTime
- FNumber
- ExposureProgram
- SpectralSensitivity
- ISOSpeedRatings
- OECF
- ShutterSpeedValue
- ApertureValue
- BrightnessValue
- ExposureBiasValue
- MaxApertureValue

- SubjectDistance
- MeteringMode
- LightSource
- Flash
- FocalLength
- SubjectArea
- FlashEnergy
- SpatialFrequencyResponse
- FocalPlaneXResolution
- FocalPlaneYResolution
- FocalPlaneResolutionUnit
- SubjectLocation
- ExposureIndex
- SensingMethod
- FileSource
- SceneType
- CFAPattern
- CustomRendered
- ExposureMode
- WhiteBalance
- DigitalZoomRatio
- FocalLengthIn35mmFilm
- SceneCaptureType
- GainControl
- Contrast
- Saturation
- Sharpness
- DeviceSettingDescription
- SubjectDistanceRange
- ImageUniqueID

- GPSVersionID
- GPSLatitudeRef
- GPSLatitude
- GPSLongitudeRef
- GPSLongitude
- GPSAltitudeRef
- GPSAltitude
- GPSTimeStamp
- GPSSatellites
- GPSStatus
- GPSMeasureMode
- GPSDOP
- GPSSpeedRef
- GPSSpeed
- GPSTrackRef
- GPSTrack
- GPSImgDirectionRef
- GPSImgDirection
- GPSMapDatum
- GPSDestLatitudeRef
- GPSDestLatitude
- GPSDestLongitudeRef
- GPSDestLongitude

- GPSDestBearingRef
- GPSDestBearing
- GPSDestDistanceRef
- GPSDestDistance
- GPSProcessingMethod
- GPSAreaInformation
- GPSDateStamp
- GPSDifferential

[Nach oben 469]

10.2.1.5 XPath/XQuery-Funktionen: Numerische

Die numerischen Erweiterungsfunktionen von Altova können in XPath- und XQuery-Ausdrücken verwendet werden und stellen zusätzliche Funktionen für die Verarbeitung von Daten zur Verfügung. Die Funktionen in diesem Abschnitt können mit dem **XPath 3.0- und XQuery 3.0**-Prozessor von Altova verwendet werden. Sie stehen im Zusammenhang mit XPath/XQuery zur Verfügung.

Anmerkung zur Benennung von Funktionen und zur Anwendbarkeit der Sprache

Altova-Erweiterungsfunktionen können in XPath/XQuery-Ausdrücken verwendet werden. Dadurch stehen neben den Funktionen in der Standardbibliothek der XPath-, XQuery- und XSLT-Funktionen zusätzliche Funktionen zur Verfügung. Die Altova-Erweiterungsfunktionen befinden sich im **Altova-Erweiterungsfunktions-Namespace, `http://www.altova.com/xslt-extensions`** und sind in diesem Abschnitt mit dem Präfix **`altova:`**, das als an diesen Namespace gebunden angenommen wird, gekennzeichnet. Beachten Sie, dass manche Funktionen in zukünftigen Versionen Ihres Produkts eventuell nicht mehr unterstützt werden oder dass sich das Verhalten einzelner Funktionen ändern kann. Um zu sehen, welche Altova-Erweiterungsfunktionen unterstützt werden, lesen Sie bitte die Dokumentation zur jeweiligen Release.

XPath-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	XP1 XP2 XP3.1
XSLT-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	XSLT1 XSLT2 XSLT3
XQuery-Funktionen (in XQuery-Ausdrücken in XQuery verwendet):	XQ1 XQ3.1

Funktionen zur automatischen Nummerierung

▼ generate-auto-number [altova:]

`altova:generate-auto-number(ID` *`als xs:string`*`, `**`StartsWith`** *`als xs:double`*`, `**`Increment`** *`als xs:double`*`, `**`ResetOnChange`** *`als xs:string`*`)` als `xs:integer` XP1 XP2 XQ1 XP3.1 XQ3.1

Generiert jedes Mal, wenn die Funktion aufgerufen wird, eine Zahl. Die erste Zahl, die beim ersten Aufruf der Funktion generiert wird, wird durch das Argument `StartsWith` definiert. Bei jedem erneuten Aufruf der Funktion wird eine neue Zahl generiert. Diese Zahl wird durch den im Argument `Increment` definierten Wert anhand der zuvor generierten Zahl inkrementiert. Auf diese Art erstellt die Funktion `altova:generate-auto-number` einen Zähler, dessen Name durch das Argument `ID` definiert wird und der jedes Mal, wenn die Funktion aufgerufen wird, inkrementiert wird. Wenn sich der Wert des Arguments `ResetOnChange` seit dem vorherigen Funktionsaufruf geändert hat, so wird der Wert der zu generierenden

Zahl auf den Wert `StartsWith` zurückgesetzt. Die Automatische Nummerierung kann auch mit der Funktion `altova:reset-auto-number` zurückgesetzt werden.

Beispiele

- **`altova:generate-auto-number`**`("ChapterNumber", 1, 1, "SomeString")` gibt bei jedem Aufruf der Funktion eine einzige Zahl beginnend mit `1` zurück, die bei jedem Aufruf der Funktion um `1` inkrementiert wird. Solange das vierte Argument in jedem anschließenden Aufruf `"SomeString"` bleibt, wird die Inkrementierung fortgesetzt. Wenn sich der Wert des vierten Arguments ändert, wird der Zähler (namens `ChapterNumber`) auf 1 zurückgesetzt. Der Wert von `ChapterNumber` kann auch folgendermaßen durch Aufruf der Funktion `altova:reset-auto-number` zurückgesetzt werden: `altova:reset-auto-number("ChapterNumber")`.

▼ reset-auto-number [altova:]

`altova:reset-auto-number(ID` *`als xs:string`***`)`** **XP1** **XP2** **XQ1** **XP3.1** **XQ3.1**

Diese Funktion setzt die Zahl des im `ID`-Argument angegebenen Zählers zur automatischen Nummerierung zurück. Die Zahl wird auf die Zahl zurückgesetzt, die durch das Argument `StartsWith` der Funktion `altova:generate-auto-number`, die den im `ID`-Argument genannten Zähler erstellt hat, definiert ist

Beispiele

- **`altova:reset-auto-number`**`("ChapterNumber")` setzt die Zahl des Zählers zur automatischen Nummerierung (`ChapterNumber`), der durch die Funktion `altova:generate-auto-number` erstellt wurde, zurück. Die Zahl wird auf den Wert des Arguments `StartsWith` der Funktion `altova:generate-auto-number`, die `ChapterNumber` erstellt hat, zurückgesetzt.

[Nach oben 474]

Numerische Funktionen

▼ hex-string-to-integer [altova:]

`altova:hex-string-to-integer(HexString` *`als xs:string`***`)`** als **`xs:integer`** **XP3.1** **XQ3.1**

Verwendet ein String-Argument, das das Base-16-Äquivalent einer Ganzzahl im Dezimalsystem (Base-10) ist, und gibt die dezimale Ganzzahl zurück.

Beispiele

- **`altova:hex-string-to-integer`**`('1')` gibt `1` zurück
- **`altova:hex-string-to-integer`**`('9')` gibt `9` zurück
- **`altova:hex-string-to-integer`**`('A')` gibt `10` zurück
- **`altova:hex-string-to-integer`**`('B')` gibt `11` zurück
- **`altova:hex-string-to-integer`**`('F')` gibt `15` zurück
- **`altova:hex-string-to-integer`**`('G')` gibt einen Fehler zurück
- **`altova:hex-string-to-integer`**`('10')` gibt `16` zurück
- **`altova:hex-string-to-integer`**`('01')` gibt `1` zurück
- **`altova:hex-string-to-integer`**`('20')` gibt `32` zurück
- **`altova:hex-string-to-integer`**`('21')` gibt `33` zurück
- **`altova:hex-string-to-integer`**`('5A')` gibt `90` zurück
- **`altova:hex-string-to-integer`**`('USA')` gibt einen Fehler zurück

integer-to-hex-string [altova:]

altova:integer-to-hex-string(Integer *as xs:integer*) **as xs:string** XP3.1 XQ3.1

Verwendet ein Ganzzahlargument und gibt das Base-16-Äquivalent als String zurück.

Beispiele

- **altova:integer-to-hex-string**(1) gibt 1 zurück
- **altova:integer-to-hex-string**(9) gibt '9' zurück
- **altova:integer-to-hex-string**(10) gibt 'A' zurück
- **altova:integer-to-hex-string**(11) gibt 'B' zurück
- **altova:integer-to-hex-string**(15) gibt 'F' zurück
- **altova:integer-to-hex-string**(16) gibt '10' zurück
- **altova:integer-to-hex-string**(32) gibt '20' zurück
- **altova:integer-to-hex-string**(33) gibt '21' zurück
- **altova:integer-to-hex-string**(90) gibt '5A' zurück

[Nach oben 474]

[nach oben 474]

10.2.1.6 XPath/XQuery-Funktionen: Schema

Die unten aufgelisteten Altova-Erweiterungsfunktionen geben Schemainformationen zurück. Weiter unten finden Sie Beschreibungen der Funktionen zusammen mit (i) Beispielen und (ii) einer Liste von Schemakomponenten und den dazugehörigen Eigenschaften. Diese Funktionen können mit dem **XPath 3.0-** und dem **XQuery 3.0-**Prozessor von Altova verwendet werden und stehen im Zusammenhang mit XPath/XQuery zur Verfügung.

Schemainformationen aus Schema-Dokumenten

Die Funktion **altova:schema** hat zwei Argumente: eines mit null Argumenten und das andere mit zwei Argumenten. Die Funktion mit null Argumenten gibt das gesamte Schema zurück. Sie können anschließend von diesem Ausgangspunkt aus durch das Schema navigieren und zu den gewünschten Schemakomponenten gehen. Die Funktion mit zwei Argumenten gibt eine bestimmte, durch Ihren QName identifizierte Komponentenart zurück. Der Rückgabewert ist in beiden Fällen eine Funktion. Um durch die zurückgegebene Komponente zu navigieren, müssen Sie eine Eigenschaft dieser spezifischen Komponente auswählen. Wenn es sich bei der Eigenschaft um ein nicht atomares Element handelt (d.h. wenn es sich um eine Komponente handelt), können Sie weiter navigieren, indem Sie eine Eigenschaft dieser Komponente auswählen. Wenn es sich bei der ausgewählten Eigenschaft um ein atomares Element handelt, wird der Wert des Elements zurückgegeben und Sie können nicht weiter navigieren.

Anmerkung: In XPath-Ausdrücken muss das Schema in die Verarbeitungsumgebung, z.B. in XSLT mit der Anweisung **xslt:import-schema** importiert werden. In XQuery-Ausdrücken muss das Schema explizit mit Hilfe eines Schemaimports importiert werden.

Schemainformationen aus XML-Nodes

Die Funktion **altova:type** übermittelt den Node eines XML-Dokuments und gibt die Typinformationen des Node aus dem PSVI zurück.

Anmerkung zur Benennung von Funktionen und zur Anwendbarkeit der Sprache

Altova-Erweiterungsfunktionen können in XPath/XQuery-Ausdrücken verwendet werden. Dadurch stehen neben den Funktionen in der Standardbibliothek der XPath-, XQuery- und XSLT-Funktionen zusätzliche Funktionen zur Verfügung. Die Altova-Erweiterungsfunktionen befinden sich im **Altova-Erweiterungsfunktions-Namespace, `http://www.altova.com/xslt-extensions`** und sind in diesem Abschnitt mit dem Präfix **`altova:`**, das als an diesen Namespace gebunden angenommen wird, gekennzeichnet. Beachten Sie, dass manche Funktionen in zukünftigen Versionen Ihres Produkts eventuell nicht mehr unterstützt werden oder dass sich das Verhalten einzelner Funktionen ändern kann. Um zu sehen, welche Altova-Erweiterungsfunktionen unterstützt werden, lesen Sie bitte die Dokumentation zur jeweiligen Release.

XPath-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	XP1 XP2 XP3.1
XSLT-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	XSLT1 XSLT2 XSLT3
XQuery-Funktionen (in XQuery-Ausdrücken in XQuery verwendet):	XQ1 XQ3.1

Schema (null Argumente)

`altova:schema() als (function(xs:string) als item()*)?` XP3.1 XQ3.1
Gibt die **`schema`**-Komponente als ganzes zurück. Durch Auswahl einer der Eigenschaften der **`schema`**-Komponente können Sie weiter in die **`schema`**-Komponente navigieren.

- Wenn es sich bei dieser Eigenschaft um eine Komponente handelt, können Sie durch Auswahl einer dieser Komponenteneigenschaften einen Schritt tiefer navigieren. Dieser Schritt kann wiederholt werden, um weiter in das Schema zu navigieren.
- Wenn es sich bei der Komponente um einen atomaren Wert handelt, wird der atomare Wert zurückgegeben und Sie können nicht tiefer navigieren.

Die Eigenschaften der **`schema`**-Komponente sind:

```
"type definitions"
"attribute declarations"
"element declarations"
"attribute group definitions"
"model group definitions"
"notation declarations"
"identity-constraint definitions"
```

Die Eigenschaften aller anderen Komponentenarten (neben **`schema`**) sind unten aufgelistet.

Anmerkung: In XQuery-Ausdrücken muss das Schema explizit importiert werden. In XPath-Ausdrücken muss das Schema in die Verarbeitungsumgebung, z.B. in XSLT mit der Anweisung **`xslt:import`** importiert worden sein.

Beispiele

- `import schema "" at "C:\Test\ExpReport.xsd"; for $typedef in altova:schema() ("type definitions") return $typedef ("name")` gibt die Namen aller simpleTypes oder complexTypes im Schema zurück.

- `import schema "" at "C:\Test\ExpReport.xsd";`

`altova:schema() ("type definitions")[1]("name")` gibt den Namen des ersten aller simpleTypes oder complexTyps im Schema zurück.

Komponenten und ihre Eigenschaften

- Assertion

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Assertion"
test	XPath-Eigenschaftsdatensatz	

- Attribute Declaration

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Attribute Declaration"
name	String	Lokaler Name des Attributs
target namespace	String	Namespace URI des Attributs
type definition	SimpleType oder ComplexType	
scope	Eine Funktion mit Eigenschaften ("class":"Scope", "variety": "global" oder "local", "parent": der enthaltende ComplexTyp bzw. die enthaltende Attributgruppe)	
value constraint	Falls vorhanden, eine Funktion mit Eigenschaften ("class": "Value Constraint", "variety": "fixed" oder "default", "value": atomarer Wert, "lexical form": String. Beachten Sie, dass die Eigenschaft "value" für Namespace-sensitive Typen nicht zur Verfügung steht.	
inheritable	Boolean	

- Attribute Group Declaration

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Attribute Group Definition"
name	String	Lokaler Name der Attributgruppe
target namespace	String	Namespace URI der Attributgruppe
attribute uses	Sequenz von (Attribute Use)	
attribute wildcard	Optionale Attribut-Wildcard	

- Attribute Use

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Attribute Use"
required	Boolean	true, wenn das Attribut obligatorisch ist, false, wenn es optional ist.
value constraint	Siehe Attribute Declaration	
inheritable	Boolean	

- Attribute Wildcard

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	string	"Wildcard"
namespace constraint	Funktion mit Eigenschaften ("class": "Namespace Constraint", "variety": "any"|"enumeration"|"not", "namespaces": Sequenz von xs:anyURI, "disallowed names": Liste mit QNames und/oder den Strings "defined" und "definedSiblings"	
process contents	String ("strict"|"lax"|"skip")	

- Complex Type

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Complex Type"
name	String	Lokaler Name des Typs (leer, wenn anonym)
target namespace	String	Namespace URI des Typs (leer, wenn anonym)
base type definition	Complex Type Definition	
final	String-Sequenz ("restriction"|"extension")	
context	Leere Sequenz (nicht implementiert)	
derivation method	String ("restriction"|"extension")	
abstract	Boolean	
attribute uses	Attribute Use-Sequenz	
attribute wildcard	Optionale Attribute Wildcard	
content type	Funktion mit Eigenschaften: ("class":"Content Type", "variety":string ("element-only"|"empty"|"mixed"|"simple"), particle:	

	optionales Partikel, "open content": Funktion mit Eigenschaften ("class":"Open Content", "mode": string ("interleave"|"suffix"), "wildcard": Wildcard), "simple type definition": Simple Type)	
prohibited substitutions	String-Sequenz ("restriction"|"extension")	
assertions	Assertion-Sequenz	

Element Declaration

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Complex Type"
name	String	Lokaler Name des Typs (leer, wenn anonym)
target namespace	String	Namespace URI des Typs (leer, wenn anonym)
type definition	Simple Type oder Complex Type	
type table	Funktion mit Eigenschaften ("class":"Type Table", "alternatives": Type Alternative-Sequenz, "default type definition": Simple Type oder Complex Type)	
scope	Funktion mit Eigenschaften ("class":"Scope", "variety": ("global"|"local"), "parent": optionaler Complex Type)	
value constraint	siehe Attribute Declaration	
nillable	Boolean	
identity-constraint definitions	Identity Constraint-Sequenz	
substitution group affiliations	Element Declaration-Sequenz	
substitution group exclusions	String-Sequenz ("restriction"|"extension")	
disallowed substitutions	String-Sequenz ("restriction"|"extension"|"substitution")	
abstract	Boolean	

Element Wildcard

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert

kind	String	"Wildcard"
namespace constraint	Funktion mit Eigenschaften ("class": "Namespace Constraint", "variety": "any"|"enumeration"|"not", "namespaces": xs:anyURI-Sequenz, "disallowed names": Liste mit QNames und/oder den Strings "defined" und "definedSiblings"	
process contents	String ("strict"|"lax"|"skip")	

Facet

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	Der Name des Facet, z.B. "minLength" oder "enumeration"
value	abhängig vom Facet	Der Wert des Facet
fixed	Boolean	
typed-value	Nur für das Enumeration Facet, Array(xs:anyAtomicType*)	Ein Array, das Enumeration-Werte, von denen jeder im Allgemeinen eine Sequenz atomarer Werte sein kann, enthält. (Anmerkung: Die Eigenschaft "value" ist unabhängig vom tatsächlichen Typ für das Enumeration Facet eine String-Sequenz)

Identity Constraint

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Identity-Constraint Definition"
name	String	Lokaler Name des Constraint
target namespace	String	Namespace URI des Constraint
identity-constraint category	String ("key"|"unique"|"keyRef")	
selector	XPath-Eigenschaftsdatensatz	
fields	Sequenz von XPath-Eigenschaftsdatensätzen	
referenced key	(nur für keyRef): Identity Constraint	Der entsprechende Key Constraint

Model Group

Property name	Eigenschaftstyp	Eigenschaftswert
kind	String	"Model Group"

compositor	String ("sequence"|"choice"|"all")	
particles	Partikel-Sequenz	

Model Group Definition

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Model Group Definition"
name	String	Lokaler Name der Model Group
target namespace	String	Namespace URI der Model Group
model group	Model Group	

Notation

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Notation Declaration"
name	String	Lokaler Name der Notation
target namespace	String	Namespace URI der Notation
system identifier	anyURI	
public identifier	String	

Particle

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Particle"
min occurs	Integer	
max occurs	Integer oder String("unbounded")	
term	Element Declaration, Element Wildcard oder ModelGroup	

Simple Type

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Simple Type Definition"
name	String	Lokaler Name des Typs (leer, wenn anonym)
target namespace	String	Namespace URI des Typs (leer, wenn anonym)
final	String-Sequenz("restriction"|"extension"|"list"|"union")	
context	enthaltende Komponente	

base type definition	Simple Type	
facets	Facet-Sequenz	
fundamental facets	Leere Sequenz (nicht implementiert)	
variety	String ("atomic"|"list"|"union")	
primitive type definition	Simple Type	
item type definition	(nur für Listentypen) Simple Type	
member type definitions	(nur für Union-Typen) Simple Type-Sequenz	

Type Alternative

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Type Alternative"
test	XPath-Eigenschaftsdatensatz	
type definition	Simple Type oder Complex Type	

XPath Property Record

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
namespace bindings	Sequenz von Funktionen mit Eigenschaften ("prefix": string, "namespace": anyURI)	
default namespace	anyURI	
base URI	anyURI	Die statische Basis-UI des XPath-Ausdrucks
expression	String	Der XPath-Ausdruck als String

Schema (zwei Argumente)

`altova:schema(ComponentKind` *`als xs:string`*`, `**`Name`** *`als xs:QName`*`) als (function(xs:string) als item()*)? XP3.1 XQ3.1`

Gibt die im ersten Argument angegebene Komponentenart zurück, welche einen Namen hat, der mit dem im zweiten Argument angegebenen Namen übereinstimmt. Durch Auswahl einer der Eigenschaften der Komponente können Sie weiter navigieren.

- Wenn es sich bei dieser Eigenschaft um eine Komponente handelt, können Sie durch Auswahl einer dieser Komponenteneigenschaften einen Schritt tiefer navigieren. Dieser Schritt kann wiederholt werden, um weiter in das Schema zu navigieren.
- Wenn es sich bei der Komponente um einen atomaren Wert handelt, wird der atomare Wert zurückgegeben und Sie können nicht tiefer navigieren.

Anmerkung: In XQuery-Ausdrücken muss das Schema explizit importiert werden. In XPath-Ausdrücken muss das Schema in die Verarbeitungsumgebung, z.B. in XSLT mit der Anweisung **`xslt:import`**

importiert worden sein.

Beispiele

- ```
  import schema "" at "C:\Test\ExpReport.xsd";
  altova:schema("element declaration", xs:QName("OrgChart"))("type definition")
  ("content type")("particles")[3]!.("term")("kind")
  ```
  gibt die `kind`-Eigenschaft des Terms der dritten `particles`-Komponente zurück. Diese particles-Komponente ist ein Nachfahr der Element-Deklaration mit dem QName `QName OrgChart`.

- ```
  import schema "" at "C:\Test\ExpReport.xsd";
  let $typedef := altova:schema("type definition", xs:QName("emailType"))
  for $facet in $typedef ("facets")
  return [$facet ("kind"), $facet("value")]
  ```
  gibt für jedes **`facet`** jeder **`emailType`**-Komponente ein Array mit der Art und dem Wert des Facet zurück.

Komponenten und ihre Eigenschaften

Assertion

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Assertion"
test	XPath-Eigenschaftsdatensatz	

Attribute Declaration

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Attribute Declaration"
name	String	Lokaler Name des Attributs
target namespace	String	Namespace URI des Attributs
type definition	SimpleType oder ComplexType	
scope	Eine Funktion mit Eigenschaften ("class":"Scope", "variety": "global" oder "local", "parent": der enthaltende ComplexTyp bzw. die enthaltende Attributgruppe)	
value constraint	Falls vorhanden, eine Funktion mit Eigenschaften ("class": "Value Constraint", "variety": "fixed" oder "default", "value": atomarer Wert, "lexical form": String. Beachten Sie, dass die Eigenschaft "value" für Namespace-sensitive Typen nicht zur Verfügung steht.	
inheritable	Boolean	

- Attribute Group Declaration

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Attribute Group Definition"
name	String	Lokaler Name der Attributgruppe
target namespace	String	Namespace URI der Attributgruppe
attribute uses	Sequenz von (Attribute Use)	
attribute wildcard	Optionale Attribut-Wildcard	

- Attribute Use

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Attribute Use"
required	Boolean	true, wenn das Attribut obligatorisch ist, false, wenn es optional ist.
value constraint	Siehe Attribute Declaration	
inheritable	Boolean	

- Attribute Wildcard

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	string	"Wildcard"
namespace constraint	Funktion mit Eigenschaften ("class": "Namespace Constraint", "variety": "any"|"enumeration"|"not", "namespaces": Sequenz von xs:anyURI, "disallowed names": Liste mit QNames und/oder den Strings "defined" und "definedSiblings"	
process contents	String ("strict"|"lax"|"skip")	

- Complex Type

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Complex Type"
name	String	Lokaler Name des Typs (leer, wenn anonym)
target namespace	String	Namespace URI des Typs (leer, wenn anonym)
base type definition	Complex Type Definition	
final	String-Sequenz ("restriction"|"extension")	

context	Leere Sequenz (nicht implementiert)	
derivation method	String ("restriction"|"extension")	
abstract	Boolean	
attribute uses	Attribute Use-Sequenz	
attribute wildcard	Optionale Attribute Wildcard	
content type	Funktion mit Eigenschaften: ("class":"Content Type", "variety":string ("element-only"|"empty"|"mixed"|"simple"), particle: optionales Partikel, "open content": Funktion mit Eigenschaften ("class":"Open Content", "mode": string ("interleave"|"suffix"), "wildcard": Wildcard), "simple type definition": Simple Type)	
prohibited substitutions	String-Sequenz ("restriction"|"extension")	
assertions	Assertion-Sequenz	

Element Declaration

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Complex Type"
name	String	Lokaler Name des Typs (leer, wenn anonym)
target namespace	String	Namespace URI des Typs (leer, wenn anonym)
type definition	Simple Type oder Complex Type	
type table	Funktion mit Eigenschaften ("class":"Type Table", "alternatives": Type Alternative-Sequenz, "default type definition": Simple Type oder Complex Type)	
scope	Funktion mit Eigenschaften ("class":"Scope", "variety": ("global"|"local"), "parent": optionaler Complex Type)	
value constraint	siehe Attribute Declaration	
nillable	Boolean	
identity-constraint definitions	Identity Constraint-Sequenz	
substitution group affiliations	Element Declaration-Sequenz	

substitution group exclusions	String-Sequenz ("restriction"|"extension")	
disallowed substitutions	String-Sequenz ("restriction"|"extension"|"substitution")	
abstract	Boolean	

- Element Wildcard

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Wildcard"
namespace constraint	Funktion mit Eigenschaften ("class": "Namespace Constraint", "variety": "any"|"enumeration"|"not", "namespaces": xs:anyURI-Sequenz, "disallowed names": Liste mit QNames und/oder den Strings "defined" und "definedSiblings"	
process contents	String ("strict"|"lax"|"skip")	

- Facet

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	Der Name des Facet, z.B. "minLength" oder "enumeration"
value	abhängig vom Facet	Der Wert des Facet
fixed	Boolean	
typed-value	Nur für das Enumeration Facet, Array(xs:anyAtomicType*)	Ein Array, das Enumeration-Werte, von denen jeder im Allgemeinen eine Sequenz atomarer Werte sein kann, enthält. (Anmerkung: Die Eigenschaft "value" ist unabhängig vom tatsächlichen Typ für das Enumeration Facet eine String-Sequenz)

- Identity Constraint

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Identity-Constraint Definition"
name	String	Lokaler Name des Constraint
target namespace	String	Namespace URI des Constraint
identity-constraint category	String ("key"|"unique"|"keyRef")	
selector	XPath-Eigenschaftsdatensatz	

fields	Sequenz von XPath-Eigenschaftsdatensätzen	
referenced key	(nur für keyRef): Identity Constraint	Der entsprechende Key Constraint

- Model Group

Property name	Eigenschaftstyp	Eigenschaftswert
kind	String	"Model Group"
compositor	String ("sequence"|"choice"|"all")	
particles	Partikel-Sequenz	

- Model Group Definition

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Model Group Definition"
name	String	Lokaler Name der Model Group
target namespace	String	Namespace URI der Model Group
model group	Model Group	

- Notation

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Notation Declaration"
name	String	Lokaler Name der Notation
target namespace	String	Namespace URI der Notation
system identifier	anyURI	
public identifier	String	

- Particle

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Particle"
min occurs	Integer	
max occurs	Integer oder String("unbounded")	
term	Element Declaration, Element Wildcard oder ModelGroup	

- Simple Type

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Simple Type Definition"

name	String	Lokaler Name des Typs (leer, wenn anonym)
target namespace	String	Namespace URI des Typs (leer, wenn anonym)
final	String-Sequenz("restriction"|"extension"|"list"|"union")	
context	enthaltende Komponente	
base type definition	Simple Type	
facets	Facet-Sequenz	
fundamental facets	Leere Sequenz (nicht implementiert)	
variety	String ("atomic"|"list"|"union")	
primitive type definition	Simple Type	
item type definition	(nur für Listentypen) Simple Type	
member type definitions	(nur für Union-Typen) Simple Type-Sequenz	

Type Alternative

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Type Alternative"
test	XPath-Eigenschaftsdatensatz	
type definition	Simple Type oder Complex Type	

XPath Property Record

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
namespace bindings	Sequenz von Funktionen mit Eigenschaften ("prefix": string, "namespace": anyURI)	
default namespace	anyURI	
base URI	anyURI	Die statische Basis-UI des XPath-Ausdrucks
expression	String	Der XPath-Ausdruck als String

Type

`altova:type(Node` *`als item?`*`)` **`als (function(xs:string) als item()*)?`** **XP3.1** **XQ3.1**

Die Funktion **`altova:type`** übermittelt einen Element- oder Attribut-Node eines XML-Dokuments und gibt die Typinformationen des Node aus dem PSVI zurück.

Anmerkung: Das XML-Dokument muss eine Schema-Deklaration haben, damit das Schema referenziert werden kann.

Beispiele

- ```
  for $element in //Email
  let $type := altova:type($element)
  return $type
  ```
  gibt eine Funktion zurück, die die Typinformationen des Node `Email` enthält.

- ```
  for $element in //Email
  let $type := altova:type($element)
  return $type ("kind")
  ```
  ermittelt anhand der Typ-Komponente des Node (Simple Type oder Complex Type) den Wert der Eigenschaft **`kind`** der Komponente.

Der Parameter **`"_props"`** gibt die Eigenschaften der ausgewählten Komponente zurück, z.B:

- ```
  for $element in //Email
  let $type := altova:type($element)
  return ($type ("kind"), $type ("_props"))
  ```
  nimmt die Typkomponente des Node `Email` (Simple Type oder Complex Type) und gibt (i) den Wert der Eigenschaft **`kind`** der Komponente zurück und anschließend (ii) die Eigenschaften dieser Komponente.

Komponenten und ihre Eigenschaften

Assertion

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Assertion"
test	XPath-Eigenschaftsdatensatz	

Attribute Declaration

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Attribute Declaration"
name	String	Lokaler Name des Attributs
target namespace	String	Namespace URI des Attributs
type definition	SimpleType oder ComplexType	
scope	Eine Funktion mit Eigenschaften ("class":"Scope", "variety": "global" oder "local", "parent": der enthaltende ComplexTyp bzw. die enthaltende Attributgruppe)	
value constraint	Falls vorhanden, eine Funktion mit Eigenschaften ("class": "Value	


  ```

| | | |
|---|---|---|
| | Constraint", "variety": "fixed" oder "default", "value": atomarer Wert, "lexical form": String. Beachten Sie, dass die Eigenschaft "value" für Namespace-sensitive Typen nicht zur Verfügung steht. | |
| inheritable | Boolean | |

- Attribute Group Declaration

| Eigenschaftsname | Eigenschaftstyp | Eigenschaftswert |
|---|---|---|
| kind | String | "Attribute Group Definition" |
| name | String | Lokaler Name der Attributgruppe |
| target namespace | String | Namespace URI der Attributgruppe |
| attribute uses | Sequenz von (Attribute Use) | |
| attribute wildcard | Optionale Attribut-Wildcard | |

- Attribute Use

| Eigenschaftsname | Eigenschaftstyp | Eigenschaftswert |
|---|---|---|
| kind | String | "Attribute Use" |
| required | Boolean | true, wenn das Attribut obligatorisch ist, false, wenn es optional ist. |
| value constraint | Siehe Attribute Declaration | |
| inheritable | Boolean | |

- Attribute Wildcard

| Eigenschaftsname | Eigenschaftstyp | Eigenschaftswert |
|---|---|---|
| kind | string | "Wildcard" |
| namespace constraint | Funktion mit Eigenschaften ("class": "Namespace Constraint", "variety": "any"\|"enumeration"\|"not", "namespaces": Sequenz von xs:anyURI, "disallowed names": Liste mit QNames und/oder den Strings "defined" und "definedSiblings" | |
| process contents | String ("strict"\|"lax"\|"skip") | |

- Complex Type

| Eigenschaftsname | Eigenschaftstyp | Eigenschaftswert |
|---|---|---|

| | | |
|---|---|---|
| kind | String | "Complex Type" |
| name | String | Lokaler Name des Typs (leer, wenn anonym) |
| target namespace | String | Namespace URI des Typs (leer, wenn anonym) |
| base type definition | Complex Type Definition | |
| final | String-Sequenz ("restriction"\|"extension") | |
| context | Leere Sequenz (nicht implementiert) | |
| derivation method | String ("restriction"\|"extension") | |
| abstract | Boolean | |
| attribute uses | Attribute Use-Sequenz | |
| attribute wildcard | Optionale Attribute Wildcard | |
| content type | Funktion mit Eigenschaften: ("class":"Content Type", "variety":string ("element-only"\|"empty"\|"mixed"\|"simple"), particle: optionales Partikel, "open content": Funktion mit Eigenschaften ("class":"Open Content", "mode": string ("interleave"\|"suffix"), "wildcard": Wildcard), "simple type definition": Simple Type) | |
| prohibited substitutions | String-Sequenz ("restriction"\|"extension") | |
| assertions | Assertion-Sequenz | |

Element Declaration

| Eigenschaftsname | Eigenschaftstyp | Eigenschaftswert |
|---|---|---|
| kind | String | "Complex Type" |
| name | String | Lokaler Name des Typs (leer, wenn anonym) |
| target namespace | String | Namespace URI des Typs (leer, wenn anonym) |
| type definition | Simple Type oder Complex Type | |
| type table | Funktion mit Eigenschaften ("class":"Type Table", "alternatives": Type Alternative-Sequenz, "default type definition": Simple Type oder Complex Type) | |
| scope | Funktion mit Eigenschaften ("class":"Scope", "variety": | |

| | | |
|---|---|---|
| | ("global"\|"local"), "parent": optionaler Complex Type) | |
| value constraint | siehe Attribute Declaration | |
| nillable | Boolean | |
| identity-constraint definitions | Identity Constraint-Sequenz | |
| substitution group affiliations | Element Declaration-Sequenz | |
| substitution group exclusions | String-Sequenz ("restriction"\|"extension") | |
| disallowed substitutions | String-Sequenz ("restriction"\|"extension"\|"substitution") | |
| abstract | Boolean | |

Element Wildcard

| Eigenschaftsname | Eigenschaftstyp | Eigenschaftswert |
|---|---|---|
| kind | String | "Wildcard" |
| namespace constraint | Funktion mit Eigenschaften ("class": "Namespace Constraint", "variety": "any"\|"enumeration"\|"not", "namespaces": xs:anyURI-Sequenz, "disallowed names": Liste mit QNames und/oder den Strings "defined" und "definedSiblings" | |
| process contents | String ("strict"\|"lax"\|"skip") | |

Facet

| Eigenschaftsname | Eigenschaftstyp | Eigenschaftswert |
|---|---|---|
| kind | String | Der Name des Facet, z.B. "minLength" oder "enumeration" |
| value | abhängig vom Facet | Der Wert des Facet |
| fixed | Boolean | |
| typed-value | Nur für das Enumeration Facet, Array(xs:anyAtomicType*) | Ein Array, das Enumeration-Werte, von denen jeder im Allgemeinen eine Sequenz atomarer Werte sein kann, enthält. (Anmerkung: Die Eigenschaft "value" ist unabhängig vom tatsächlichen Typ für das Enumeration Facet eine String-Sequenz) |

Identity Constraint

| Eigenschaftsname | Eigenschaftstyp | Eigenschaftswert |
|---|---|---|
| kind | String | "Identity-Constraint Definition" |
| name | String | Lokaler Name des Constraint |
| target namespace | String | Namespace URI des Constraint |
| identity-constraint category | String ("key"\|"unique"\|"keyRef") | |
| selector | XPath-Eigenschaftsdatensatz | |
| fields | Sequenz von XPath-Eigenschaftsdatensätzen | |
| referenced key | (nur für keyRef): Identity Constraint | Der entsprechende Key Constraint |

Model Group

| Property name | Eigenschaftstyp | Eigenschaftswert |
|---|---|---|
| kind | String | "Model Group" |
| compositor | String ("sequence"\|"choice"\|"all") | |
| particles | Partikel-Sequenz | |

Model Group Definition

| Eigenschaftsname | Eigenschaftstyp | Eigenschaftswert |
|---|---|---|
| kind | String | "Model Group Definition" |
| name | String | Lokaler Name der Model Group |
| target namespace | String | Namespace URI der Model Group |
| model group | Model Group | |

Notation

| Eigenschaftsname | Eigenschaftstyp | Eigenschaftswert |
|---|---|---|
| kind | String | "Notation Declaration" |
| name | String | Lokaler Name der Notation |
| target namespace | String | Namespace URI der Notation |
| system identifier | anyURI | |
| public identifier | String | |

Particle

| Eigenschaftsname | Eigenschaftstyp | Eigenschaftswert |
|---|---|---|
| kind | String | "Particle" |
| min occurs | Integer | |

| | | |
|---|---|---|
| max occurs | Integer oder String("unbounded") | |
| term | Element Declaration, Element Wildcard oder ModelGroup | |

Simple Type

| Eigenschaftsname | Eigenschaftstyp | Eigenschaftswert |
|---|---|---|
| kind | String | "Simple Type Definition" |
| name | String | Lokaler Name des Typs (leer, wenn anonym) |
| target namespace | String | Namespace URI des Typs (leer, wenn anonym) |
| final | String-Sequenz("restriction"\|"extension"\|"list"\|"union") | |
| context | enthaltende Komponente | |
| base type definition | Simple Type | |
| facets | Facet-Sequenz | |
| fundamental facets | Leere Sequenz (nicht implementiert) | |
| variety | String ("atomic"\|"list"\|"union") | |
| primitive type definition | Simple Type | |
| item type definition | (nur für Listentypen) Simple Type | |
| member type definitions | (nur für Union-Typen) Simple Type-Sequenz | |

Type Alternative

| Eigenschaftsname | Eigenschaftstyp | Eigenschaftswert |
|---|---|---|
| kind | String | "Type Alternative" |
| test | XPath-Eigenschaftsdatensatz | |
| type definition | Simple Type oder Complex Type | |

XPath Property Record

| Eigenschaftsname | Eigenschaftstyp | Eigenschaftswert |
|---|---|---|
| namespace bindings | Sequenz von Funktionen mit Eigenschaften ("prefix": string, "namespace": anyURI) | |
| default namespace | anyURI | |
| base URI | anyURI | Die statische Basis-UI des XPath-Ausdrucks |

| | | |
|---|---|---|
| expression | String | Der XPath-Ausdruck als String |

## 10.2.1.7 XPath/XQuery-Funktionen: Sequenz

Die Sequenz-Erweiterungsfunktionen von Altova können in XPath- und XQuery-Ausdrücken verwendet werden und stellen zusätzliche Funktionen für die Verarbeitung von Daten zur Verfügung. Die Funktionen in diesem Abschnitt können mit dem **XPath 3.0-** und **XQuery 3.0**-Prozessor von Altova verwendet werden. Sie stehen im Zusammenhang mit XPath/XQuery zur Verfügung.

Anmerkung zur Benennung von Funktionen und zur Anwendbarkeit der Sprache

Altova-Erweiterungsfunktionen können in XPath/XQuery-Ausdrücken verwendet werden. Dadurch stehen neben den Funktionen in der Standardbibliothek der XPath-, XQuery- und XSLT-Funktionen zusätzliche Funktionen zur Verfügung. Die Altova-Erweiterungsfunktionen befinden sich im **Altova-Erweiterungsfunktions-Namespace, `http://www.altova.com/xslt-extensions`** und sind in diesem Abschnitt mit dem Präfix **`altova:`**, das als an diesen Namespace gebunden angenommen wird, gekennzeichnet. Beachten Sie, dass manche Funktionen in zukünftigen Versionen Ihres Produkts eventuell nicht mehr unterstützt werden oder dass sich das Verhalten einzelner Funktionen ändern kann. Um zu sehen, welche Altova-Erweiterungsfunktionen unterstützt werden, lesen Sie bitte die Dokumentation zur jeweiligen Release.

| | |
|---|---|
| *XPath-Funktionen (in XPath-Ausdrücken in XSLT verwendet):* | XP1 XP2 XP3.1 |
| *XSLT-Funktionen (in XPath-Ausdrücken in XSLT verwendet):* | XSLT1 XSLT2 XSLT3 |
| *XQuery-Funktionen (in XQuery-Ausdrücken in XQuery verwendet):* | XQ1 XQ3.1 |

▼ attributes [altova:]

**`altova:attributes(AttributeName`** *als xs:string*`)` als **`attribute()*`** XP3.1 XQ3.1
Gibt alle Attribute zurück, die einen lokalen Namen haben, der mit dem im Input-Argument `AttributeName` angegebenen Namen identisch ist. Die Groß- und Kleinschreibung wird bei der Suche, die entlang der `attribute::` Achse durchgeführt wird, beachtet. Das bedeutet, dass der Kontext-Node der Parent-Element-Node sein muss.

⊟ *Beispiele*

- **`altova:attributes`**`("MyAttribute")` gibt `MyAttribute()*` zurück

**`altova:attributes(AttributeName`** *als xs:string*`, SearchOptions` *als xs:string*`)` als **`attribute()*`** XP3.1 XQ3.1
Gibt alle Attribute zurück, die einen lokalen Namen haben, der mit dem im Input-Argument `AttributeName` angegebenen Namen identisch ist. Die Groß- und Kleinschreibung wird bei der Suche, die entlang der `attribute::` Achse durchgeführt wird, beachtet. Der Kontext-Node muss der Parent-Element-Node sein. Das zweite Argument ist ein String, der Options-Flags enthält. Zur Verfügung stehen die folgenden Flags:
**`r`** = wechselt zu einer Suche mittels Regular Expression; bei `AttributeName` muss es sich in diesem Fall

um einen Regular Expression-Suchstring handeln;
**f** = Wenn diese Option definiert ist, liefert `AttributeName` eine vollständige Übereinstimmung; andernfalls muss `AttributeName` nur teilweise mit einem Attributnamen übereinstimmen, damit dieses Attribut zurückgegeben wird. Wenn **f** z.B. nicht definiert ist, gibt `MyAtt MyAttribute` zurück;
**i** = wechselt zu einer Suche ohne Berücksichtigung der Groß- und Kleinschreibung;
**p** = inkludiert das Namespace-Präfix in die Suche; `AttributeName` sollte in diesem Fall das Namespace-Präfix enthalten, z.B.: `altova:MyAttribute`.
Die Flags können in jeder Reihenfolge angegeben werden. Ungültige Flags erzeugen eine Fehlermeldung. Sie können ein oder mehrere Flags weglassen. Es ist auch der leere String zulässig. Das Resultat ist dasselbe wie bei Verwendung der Funktion mit nur einem Argument (*siehe vorherige Signatur*). Unzulässig ist jedoch die Verwendung einer leeren Sequenz als zweites Argument.

*Beispiele*

- **altova:attributes**("MyAttribute", "rfip") gibt `MyAttribute()*` zurück
- **altova:attributes**("MyAttribute", "pri") gibt `MyAttribute()*` zurück
- **altova:attributes**("MyAtt", "rip") gibt `MyAttribute()*` zurück
- **altova:attributes**("MyAttributes", "rfip") gibt keine Übereinstimmung zurück
- **altova:attributes**("MyAttribute", "") gibt `MyAttribute()*` zurück
- **altova:attributes**("MyAttribute", "Rip") gibt einen Fehler zurück, dass das Flag unbekannt ist.
- **altova:attributes**("MyAttribute", ) gibt den Fehler zurück, dass das zweite Argument fehlt.

## elements [altova:]

**altova:elements(ElementName** *als xs:string*) als **element()*** **XP3.1** **XQ3.1**
Gibt alle Elemente zurück, die einen lokalen Namen haben, der mit dem im Input-Argument `ElementName` angegebenen Namen identisch ist. Die Groß- und Kleinschreibung wird bei der Suche, die entlang der `child::` Achse durchgeführt wird, beachtet. Der Kontext-Node muss der Parent-Node des gesuchten Elements sein.

*Beispiele*

- **altova:elements**("MyElement") gibt `MyElement()*` zurück

**altova:elements(ElementName** *als xs:string*, **SearchOptions** *als xs:string*) als **element()*** **XP3.1** **XQ3.1**
Gibt alle Elemente zurück, die einen lokalen Namen haben, der mit dem im Input-Argument `ElementName` angegebenen Namen identisch ist. Die Groß- und Kleinschreibung wird bei der Suche, die entlang der `child::` Achse durchgeführt wird, beachtet. Der Kontext-Node muss der Parent-Node des gesuchten Elements sein. Das zweite Argument ist ein String, der Options-Flags enthält. Zur Verfügung stehen die folgenden Flags:
**r** = wechselt zu einer Suche mittels Regular Expression; bei `ElementName` muss es sich in diesem Fall um einen Regular Expression-Suchstring handeln;
**f** = Wenn diese Option definiert ist, liefert `ElementName` eine vollständige Übereinstimmung; andernfalls muss `ElementName` nur teilweise mit einem Elementnamen übereinstimmen, damit dieses Element zurückgegeben wird. Wenn **f** z.B. nicht definiert ist, gibt `MyElem MyElement` zurück;
**i** = wechselt zu einer Suche ohne Berücksichtigung der Groß- und Kleinschreibung;
**p** = inkludiert das Namespace-Präfix in die Suche; `ElementName` sollte in diesem Fall das Namespace-Präfix enthalten, z.B.: `altova:MyElement`.
Die Flags können in jeder Reihenfolge angegeben werden. Ungültige Flags erzeugen eine Fehlermeldung. Sie können ein oder mehrere Flags weglassen. Es ist auch der leere String zulässig. Das Resultat ist dasselbe wie bei Verwendung der Funktion mit nur einem Argument (siehe vorherige Signatur). Unzulässig

ist jedoch die Verwendung einer leeren Sequenz.

*Beispiele*

- **altova:elements**("MyElement", "rip") gibt MyElement()* zurück
- **altova:elements**("MyElement", "pri") gibt MyElement()* zurück
- **altova:elements**("MyElement", "") gibt MyElement()* zurück
- **altova:elements**("MyElem", "rip") gibt MyElement()* zurück
- **altova:elements**("MyElements", "rfip") gibt keine Übereinstimmung zurück
- **altova:elements**("MyElement", "Rip") gibt einen Fehler zurück, dass das Flag unbekannt ist.
- **altova:elements**("MyElement", ) gibt den Fehler zurück, dass das zweite Argument fehlt.

## find-first [altova:]

**altova:find-first( (***item()**)**, (CheckFunction( item()** *als xs:boolean***) )** als **item()? XP3.1 XQ3.1**

Diese Funktion verwendet zwei Argumente. Das erste Argument ist eine Sequenz von einem oder mehreren Elementen eines beliebigen Datentyps. Das zweite Argument, Condition, ist eine Referenz zu einer XPath-Funktion, die ein Argument erhält. (hat einen Stellenwert 1) und einen Booleschen Wert zurückgibt. Jedes Element von **Sequence** wird der Reihe nach der in Condition referenzierten Funktion bereitgestellt. (*Beachten Sie:* Die Funktion hat ein einziges Argument.) Das erste **Sequence** Element, bei dem das Resultat von **Condition true()** ist, wird als das Ergebnis von altova:find-first zurückgegeben. Anschließend wird die Iteration gestoppt.

*Beispiele*

- **altova:find-first**(5 to 10, function($a) {$a mod 2 = 0}) gibt xs:integer 6 zurück

Das Argument **Condition** referenziert die XPath 3.0 Inline-Funktion, **function()**, welche eine Inline-Funktion **$a** deklariert und diese anschließend definiert. Die einelnen Elemente im Argument **Sequence** von **altova:find-first** werden der Reihe nach an **$a** als sein Input-Wert übergeben. Der Input-Wert wird an der Bedingung in der Funktionsdefinition ($a mod 2 = 0) überprüft. Der erste Input-Wert, der diese Bedingung erfüllt, wird als das Ergebnis von **altova:find-first** (in diese Fall **6**) zurückgegeben.

- **altova:find-first**((1 to 10), (function($a) {$a+3=7})) gibt xs:integer 4 zurück

*Weitere Beispiele*

Wenn die Datei **C:\Temp\Customers.xml** vorhanden ist:

- **altova:find-first**( ("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1) ) gibt C:\Temp\Customers.xml zurück

Wenn die Datei **C:\Temp\Customers.xml** nicht vorhanden ist und **http://www.altova.com/index.html** vorhanden ist:

- **altova:find-first**( ("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1) ) gibt http://www.altova.com/index.html zurück

Wenn weder die Datei **C:\Temp\Customers.xml** noch **http://www.altova.com/index.html** vorhanden ist:

- **altova:find-first**( ("C:\Temp\Customers.xml", "http://www.altova.com/index.html"),

`(doc-available#1) )` gibt kein Ergebnis zurück

*Anmerkungen zu den obigen Beispielen*

- Die XPath 3.0-Funktion, `doc-available`, erhält ein einziges Argument, das als URI verwendet wird. Sie gibt nur dann `true` zurück, wenn unter der angegebenen URI ein Dokument-Node gefunden wird. Das Dokument unter der angegebenen URI muss daher ein XML-Dokument sein.
- Die Funktion `doc-available` kann für **`Condition`**, das zweite Argument von `altova:find-first` verwendet werden, da sie nur ein Argument erhält (Stelligkeit=1), da sie ein Element `item()` als Input erhält (ein String, der als URI verwendet wird) und einen Booleschen Wert zurückgibt.
- Beachten Sie, dass `doc-available` nur referenziert und nicht direkt aufgerufen wird. Das angehängte Suffix `#1` gibt eine Funktion mit einer Stelligkeit 1 an. Als Ganzes bedeutet `doc-available#1`: *Verwende die Funktion doc-availabe(), welche die Stelligkeit=1 hat und übergib die einzelnen Elemente in der ersten Sequenz der Reihe nach als einziges Argument an die Funktion.* Als Ergebnis wird jeder der beiden Strings an **`doc-available()`** übergeben. Die Funktion verwendet den String als URI und überprüft, ob unter der URI ein Dokument-Node vorhanden ist. Wenn dies der Fall ist, wird **`doc-available()`** zu `true()` ausgewertet und der String wird als Ergebnis der Funktion **`altova:find-first`** zurückgegeben. *Beachten Sie zur Funktion doc-available(), dass relative Pfade relativ zu aktuellen Basis-URI aufgelöst werden. Die Basis-URI ist standardmäßig die URI des XML-Dokuments, von dem aus die Funktion geladen wird.*

## find-first-combination [altova:]

**`altova:find-first-combination((Seq-01`** *as item()**), **(Seq-02** *as item()**),
**(Condition( Seq-01-Item, Seq-02-Item** *as xs:boolean*)) **as item()*** **XP3.1** **XQ3.1**
Diese Funktion verwendet drei Argumente:

- Die ersten beiden Argumente, **`Seq-01`** und **`Seq-02`**, sind Sequenzen von einem oder mehreren Elementen eines beliebigen Datentyps.
- Das dritte Argument, **`Condition`**, ist eine Referenz auf eine XPath-Funktion, die zwei Argumente erhält (d.h. eine Stelligkeit **2** hat) und einen Booleschen Wert zurückgibt.

Die Elemente von **`Seq-01`** und **`Seq-02`** werden als die Argumente der Funktion **`Condition`** in geordneten Paaren übergeben (je ein Element aus jeder Sequenz bildet ein Paar). Die Paare sind folgendermaßen geordnet.

```
If   Seq-01 = X1, X2, X3 ... Xn
And Seq-02 = Y1, Y2, Y3 ... Yn
Then (X1 Y1), (X1 Y2), (X1 Y3) ... (X1 Yn), (X2 Y1), (X2 Y2) ... (Xn Yn)
```

Das erste geordnete Paar, bei dem die Funktion **`Condition`** zu **`true()`** ausgewertet wird, wird als Ergebnis von **`altova:find-first-combination`** zurückgegeben. Beachten Sie: (i) Wenn die Funktion **`Condition`** durch die bereitgestellten Argumentpaare iteriert und nicht ein einziges Mal zu **`true()`** ausgewertet wird, so gibt **`altova:find-first-combination`** *Keine Ergebnisse* zurück; (ii) Das Ergebnis von **`altova:find-first-combination`** ist immer ein Elementpaar (eines beliebigen Datentyps) oder gar kein Element.

Beispiele

- `altova:find-first-combination(11 to 20, 21 to 30, function($a, $b) {$a+$b = 32})` gibt die Sequenz `xs:integers (11, 21)` zurück
- `altova:find-first-combination(11 to 20, 21 to 30, function($a, $b) {$a+$b = 33})` gibt die Sequenz `xs:integers (11, 22)` zurück
- `altova:find-first-combination(11 to 20, 21 to 30, function($a, $b) {$a+$b = 34})` gibt die Sequenz `xs:integers (11, 23)` zurück

find-first-pair [altova:]

`altova:find-first-pair((Seq-01` *`als item()*`*`), (Seq-02` *`als item()*`*`), (Condition( Seq-01-Item, Seq-02-Item` *`als xs:boolean`*`)) as item()* XP3.1 XQ3.1`

Diese Funktion erhält drei Argumente:

- Die ersten beiden Argumente, **`Seq-01`** und **`Seq-02`**, sind Sequenzen von einem oder mehreren Elementen eines beliebigen Datentyps.
- Das dritte Argument, **`Condition`**, ist eine Referenz auf eine XPath-Funktion, die zwei Argumente erhält (d.h. eine Stelligkeit **`2`** hat) und einen Booleschen Wert zurückgibt.

Die Elemente von **`Seq-01`** und **`Seq-02`** werden als die Argumente der Funktion **`Condition`** in geordneten Paaren übergeben. Die Paare sind folgendermaßen geordnet.

```
If   Seq-01 = X1, X2, X3 ... Xn
And  Seq-02 = Y1, Y2, Y3 ... Yn
Then (X1 Y1), (X2 Y2), (X3 Y3) ... (Xn Yn)
```

Das erste geordnete Paar, bei dem die Funktion **`Condition`** zu **`true()`** ausgewertet wird, wird als Ergebnis von **`altova:find-first-pair`** zurückgegeben. Beachten Sie: (i) Wenn die Funktion **`Condition`** durch die bereitgestellten Argumentpaare iteriert und nicht ein einziges Mal zu **`true()`** ausgewertet wird, so gibt **`altova:find-first-pair`** *Keine Ergebnisse* zurück; (ii) Das Ergebnis von **`altova:find-first-pair`** ist immer ein Elementpaar (eines beliebigen Datentyps) oder gar kein Element.

Beispiele

- **`altova:find-first-pair`**`(11 to 20, 21 to 30, function($a, $b) {$a+$b = 32})` gibt die Sequenz `xs:integers (11, 21)` zurück
- **`altova:find-first-pair`**`(11 to 20, 21 to 30, function($a, $b) {$a+$b = 33})` gibt *Keine Ergebnisse* zurück

Beachten Sie anhand der zwei Beispiel oben, dass die Paare folgendermaßen geordnet sind: `(11, 21) (12, 22) (13, 23)...(20, 30)`. Aus diesem Grund gibt das zweite Beispiel *Keine Ergebnisse* zurück (da keine geordnetes Paar die Summe `33` ergibt).

find-first-pair-pos [altova:]

`altova:find-first-pair-pos((Seq-01` *`as item()*`*`), (Seq-02` *`as item()*`*`), (Condition( Seq-01-Item, Seq-02-Item` *`as xs:boolean`*`)) as xs:integer XP3.1 XQ3.1`

Diese Funktion erhält drei Argumente:

- Die ersten beiden Argumente, **`Seq-01`** und **`Seq-02`**, sind Sequenzen von einem oder mehreren Elementen eines beliebigen Datentyps.

- Das dritte Argument, **Condition**, ist eine Referenz auf eine XPath-Funktion, die zwei Argumente erhält (d.h. eine Stelligkeit 2 hat) und einen Booleschen Wert zurückgibt.

Die Elemente von **Seq-01** und **Seq-02** werden als die Argumente der Funktion **Condition** in geordneten Paaren übergeben. Die Paare sind folgendermaßen geordnet.

```
If   Seq-01 = X1, X2, X3 ... Xn
And  Seq-02 = Y1, Y2, Y3 ... Yn
Then (X1 Y1), (X2 Y2), (X3 Y3) ... (Xn Yn)
```

Als Ergebnis von **altova:find-first-pair-pos** wird die Indexposition des ersten geordneten Paars, bei dem die Funktion **Condition** zu **true()** ausgewertet wird, zurückgegeben. Beachten Sie: Wenn die Funktion **Condition** durch die bereitgestellten Argumentpaare iteriert und kein einziges Mal zu **true()** ausgewertet wird, so gibt **altova:find-first-pair-pos** *Keine Ergebnisse* zurück.

Beispiele

- **altova:find-first-pair-pos**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 32}) gibt 1 zurück
- **altova:find-first-pair-pos**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 33}) gibt *Keine Ergebnisse* zurück

Beachten Sie anhand der zwei Beispiel oben, dass die Paare folgendermaßen geordnet sind: (11, 21) (12, 22) (13, 23)...(20, 30). Im ersten Beispiel gibt die Funktion **Condition** bei Auswertung des ersten Paars **true()** zurück, daher wird dessen Indexposition in der Sequenz, 1, zurückgegeben. Das zweite Beispiel gibt *Keine Ergebnisse* zurück (da keine geordnetes Paar die Summe 33 ergibt).

find-first-pos [altova:]

altova:find-first-pos((*item()****), (CheckFunction(item()** *als xs:boolean***))** als **xs:integer? XP3.1 XQ3.1**

Diese Funktion verwendet zwei Argumente. Das erste Argument ist eine Sequenz von einem oder mehreren Elementen eines beliebigen Datentyps. Das zweite Argument, Condition, ist eine Referenz zu einer XPath-Funktion, die ein Argument erhält. (hat einen Stellenwert 1) und einen Booleschen Wert zurückgibt. Jedes Element von **Sequence** wird der Reihe nach der in Condition referenzierten Funktion bereitgestellt. (*Beachten Sie:* Die Funktion hat ein einziges Argument.) Das erste **Sequence** Element, bei dem das Resultat von **Condition true()** ist, wird als das Ergebnis von **altova:find-first-pos** zurückgegeben. Anschließend wird die Iteration gestoppt.

Beispiele

- **altova:find-first-pos**(5 to 10, function($a) {$a mod 2 = 0}) gibt xs:integer 2 zurück

Das Argument **Condition** referenziert die XPath 3.0 Inline-Funktion, **function()**, welche eine Inline-Funktion **$a** deklariert und diese anschließend definiert. Die einelnen Elemente im Argument **Sequence** von **altova:find-first-pos** werden der Reihe nach an **$a** als sein Input-Wert übergeben. Der Input-Wert wird an der Bedingung in der Funktionsdefinition ($a mod 2 = 0) überprüft. Die Indexposition in der Sequenz des ersten Input-Werts, die diese Bedingung erfüllt, wird als das Ergebnis von **altova:find-first-pos** zurückgegeben (in diesem Fall 2, da **6**, der erste Wert in der Sequenz, der die Bedingung erfüllt, sich in der Sequenz an der Indexposition 2 befindet).

Weitere Beispiele
Wenn die Datei **C:\Temp\Customers.xml** vorhanden ist:

- **altova:find-first-pos**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) gibt 1 zurück

Wenn die Datei **C:\Temp\Customers.xml** nicht vorhanden ist und **http://www.altova.com/index.html** vorhanden ist:

- **altova:find-first-pos**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) gibt 2 zurück

Wenn weder die Datei **C:\Temp\Customers.xml** noch **http://www.altova.com/index.html** vorhanden ist:

- **altova:find-first-pos**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) gibt kein Ergebnis zurück

Anmerkungen zu den obigen Beispielen

- Die XPath 3.0-Funktion, doc-available, erhält ein einziges Argument, das als URI verwendet wird. Sie gibt nur dann true zurück, wenn unter der angegebenen URI ein Dokument-Node gefunden wird. (Das Dokument unter der angegebenen URI muss daher ein XML-Dokument sein.)
- Die Funktion doc-available kann für **Condition,** das zweite Argument von altova:find-first-pos verwendet werden, da sie nur ein Argument erhält (Stelligkeit=1), da sie ein Element item() als Input erhält (ein String, der als URI verwendet wird) und einen Booleschen Wert zurückgibt.
- Beachten Sie, dass doc-available nur referenziert und nicht direkt aufgerufen wird. Das angehängte Suffix #1 gibt eine Funktion mit einer Stelligkeit 1 an. Als Ganzes bedeutet doc-available#1: *Verwende die Funktion doc-availabe(), welche die Stelligkeit=1 hat und übergib die einzelnen Elemente in der ersten Sequenz der Reihe nach als einziges Argument an die Funktion.* Als Ergebnis wird jeder der beiden Strings an **doc-available()** übergeben. Die Funktion verwendet den String als URI und überprüft, ob unter der URI ein Dokument-Node vorhanden ist. Wenn dies der Fall ist, wird **doc-available()** zu true() ausgewertet und der String wird als Ergebnis der Funktion **altova:find-first** zurückgegeben. *Beachten Sie zur Funktion doc-available(), dass relative Pfade relativ zu aktuellen Basis-URI aufgelöst werden. Die Basis-URI ist standardmäßig die URI des XML-Dokuments, von dem aus die Funktion geladen wird.*

for-each-attribute-pair [altova:]

altova:for-each-attribute-pair(Seq1 *als element()?*, **Seq2** *als element()?*, **Function** *als function()***)** als **item()*** **XP3.1** **XQ3.1**
Die beiden ersten Argumente identifizieren zwei Elemente, anhand deren Attribute Attributpaare gebildet werden, wobei das eine Attribut eines Paars aus dem ersten Element und das andere aus dem zweiten Element stammt. Die Attributpaare werden auf Basis ihres übereinstimmenden Namens ausgewählt und alphabetisch (nach ihren Namen) zu einer Gruppe geordnet. Falls es zu einem Attribut im anderen Element keine Entsprechung gibt, ist das Paar "nicht verbunden", d.h. es besteht nur aus einem Mitglied. Das Funktionselement (das dritte Argument Function) wird auf die einzelnen Paare (verbundene und nicht

verbundene) in der Sequenz der Paare separat angewendet, wodurch als Ausgabe eine Sequenz von Einträgen erzeugt wird.

Beispiele

- `altova:for-each-attribute-pair(/Example/Test-A, /Example/Test-B, function($a, $b) {$a+b})` gibt zurück ...

  `(2, 4, 6)` wenn
  `<Test-A att1="1" att2="2" att3="3" />`
  `<Test-B att1="1" att2="2" att3="3" />`

  `(2, 4, 6)` wenn
  `<Test-A att2="2" att1="1" att3="3" />`
  `<Test-B att3="3" att2="2" att1="1" />`

  `(2, 6)` wenn
  `<Test-A att4="4" att1="1" att3="3" />`
  `<Test-B att3="3" att2="2" att1="1" />`

  Anmerkung: *Das Ergebnis* `(2, 6)` *wird mit Hilfe der folgenden Aktion ermittelt:* `(1+1, ()+2, 3+3, 4+())`. *Wenn einer der Operanden eine leere Sequenz ist, wie dies bei Eintrag 2 und 4 der Fall ist, so ist das Ergebnis der Addition eine leere Sequenz.*

- `altova:for-each-attribute-pair(/Example/Test-A, /Example/Test-B, concat#2)` gibt zurück ...

  `(11, 22, 33)` wenn
  `<Test-A att1="1" att2="2" att3="3" />`
  `<Test-B att1="1" att2="2" att3="3" />`

  `(11, 2, 33, 4)` wenn
  `<Test-A att4="4" att1="1" att3="3" />`
  `<Test-B att3="3" att2="2" att1="1" />`

for-each-combination [altova:]

`altova:for-each-combination(FirstSequence` *`als item()*`**`, SecondSequence`** *`als item()*`**`, Function($i,$j){$i || $j} )`** als `item()*` XP3.1 XQ3.1

Die Elemente der zwei Sequenzen in den ersten beiden Argumenten werden miteinander kombiniert, so dass jedes Element in der ersten Sequenz der Reihe nach einmal mit jedem Element in der zweiten Sequenz kombiniert wird. Die als drittes Argument angegebene Funktion wird auf die einzelnen Kombinationen in der erzeugten Sequenz angewendet, wodurch als Ausgabe eine Sequenz von Elementen erzeugt wird (*siehe Beispiel*).

Beispiele

- `altova:for-each-combination( ('a', 'b', 'c'), ('1', '2', '3'), function($i, $j) {$i || $j} )` gibt `('a1', 'a2', 'a3', 'b1', 'b2', 'b3', 'c1', 'c2', 'c3')` zurück

for-each-matching-attribute-pair [altova:]

`altova:for-each-matching-attribute-pair(Seq1` *`als element()?`*`, `**`Seq2`** *`als element()?`*`,`

Function *als function()*) als **item()*** XP3.1 XQ3.1
Die beiden ersten Argumente identifizieren zwei Elemente, anhand deren Attribute Attributpaare gebildet werden, wobei das eine Attribut eines Paars aus dem ersten Element und das andere aus dem zweiten Element stammt. Die Attributpaare werden auf Basis ihres übereinstimmenden Namens ausgewählt und alphabetisch (nach ihren Namen) zu einer Gruppe geordnet. Falls es zu einem Attribut im anderen Element keine Entsprechung gibt, wird kein Paar gebildet. Das Funktionselement (das dritte Argument `Function`) wird auf die einzelnen Paare in der Sequenz der Paare separat angewendet, wodurch als Ausgabe eine Sequenz von Einträgen erzeugt wird.

Beispiele

- **altova:for-each-matching-attribute-pair**(/Example/Test-A, /Example/Test-B, function($a, $b){$a+b}) gibt zurück ...

  (2, 4, 6) wenn
  ```
  <Test-A att1="1" att2="2" att3="3" />
  <Test-B att1="1" att2="2" att3="3" />
  ```

  (2, 4, 6) wenn
  ```
  <Test-A att2="2" att1="1" att3="3" />
  <Test-B att3="3" att2="2" att1="1" />
  ```

  (2, 6) wenn
  ```
  <Test-A att4="4" att1="1" att3="3" />
  <Test-B att3="3" att2="2" att3="1" />
  ```

- **altova:for-each-matching-attribute-pair**(/Example/Test-A, /Example/Test-B, concat#2) gibt zurück ....

  (11, 22, 33) wenn
  ```
  <Test-A att1="1" att2="2" att3="3" />
  <Test-B att1="1" att2="2" att3="3" />
  ```

  (11, 33) wenn
  ```
  <Test-A att4="4" att1="1" att3="3" />
  <Test-B att3="3" att2="2" att1="1" />
  ```

substitute-empty [altova:]

altova:substitute-empty(FirstSequence *als item()**, **SecondSequence** *als item()*) als **item()*** XP3.1 XQ3.1
Wenn `FirstSequence` leer ist, wird `SecondSequence` zurückgegeben. Wenn `FirstSequence` nicht leer ist, wird `FirstSequence` zurückgegeben.

Beispiele

- **altova:substitute-empty**((1,2,3), (4,5,6)) gibt (1,2,3) zurück
- **altova:substitute-empty**((), (4,5,6)) gibt (4,5,6) zurück

10.2.1.8 XPath/XQuery-Funktionen: String

Die folgenden XPath/XQuey-Erweiterungsfunktionen für Strings werden in der aktuellen Version Ihres Altova-Produkts unterstützt und bieten Zusatzfunktionalitäten für die Verarbeitung von Daten. Die Funktionen in diesem Abschnitt können mit dem **XPath 3.0-** und **XQuery 3.0**-Prozessor von Altova verwendet werden. Sie stehen im Zusammenhang mit XPath/XQuery zur Verfügung.

Anmerkung zur Benennung von Funktionen und zur Anwendbarkeit der Sprache

Altova-Erweiterungsfunktionen können in XPath/XQuery-Ausdrücken verwendet werden. Dadurch stehen neben den Funktionen in der Standardbibliothek der XPath-, XQuery- und XSLT-Funktionen zusätzliche Funktionen zur Verfügung. Die Altova-Erweiterungsfunktionen befinden sich im **Altova-Erweiterungsfunktions-Namespace, `http://www.altova.com/xslt-extensions`** und sind in diesem Abschnitt mit dem Präfix **`altova:`**, das als an diesen Namespace gebunden angenommen wird, gekennzeichnet. Beachten Sie, dass manche Funktionen in zukünftigen Versionen Ihres Produkts eventuell nicht mehr unterstützt werden oder dass sich das Verhalten einzelner Funktionen ändern kann. Um zu sehen, welche Altova-Erweiterungsfunktionen unterstützt werden, lesen Sie bitte die Dokumentation zur jeweiligen Release.

XPath-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	XP1 XP2 XP3.1
XSLT-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	XSLT1 XSLT2 XSLT3
XQuery-Funktionen (in XQuery-Ausdrücken in XQuery verwendet):	XQ1 XQ3.1

▼ camel-case [altova:]

`altova:camel-case(InputString` *als xs:string*`)` als **`xs:string`** XP3.1 XQ3.1
Gibt den Input-String **`InputString`** in CamelCase zurück. Der String wird mit Hilfe der Regular Expression **`'\s'`** (welches ein Kürzel für das Leerzeichen ist) analysiert. Das erste Zeichen nach einem Leerzeichen oder einer Sequenz aufeinander folgender Leerzeichen, das kein Leerzeichen ist, wird mit einem Großbuchstaben geschrieben. Das erste Zeichen im Ausgabestring wird mit einem Großbuchstaben geschrieben.

Beispiele

- **`altova:camel-case`**`("max")` gibt `Max` zurück
- **`altova:camel-case`**`("max max")` gibt `Max Max` zurück
- **`altova:camel-case`**`("file01.xml")` gibt `File01.xml` zurück
- **`altova:camel-case`**`("file01.xml file02.xml")` gibt `File01.xml File02.xml` zurück
- **`altova:camel-case`**`("file01.xml  file02.xml")` gibt `File01.xml  File02.xml` zurück
- **`altova:camel-case`**`("file01.xml -file02.xml")` gibt `File01.xml -file02.xml` zurück

`altova:camel-case(InputString` *als xs:string*`,` **`SplitChars`** *als xs:string*`,` **`IsRegex`** *als xs:boolean*`)` als **`xs:string`** XP3.1 XQ3.1
Konvertiert den Input-String **`InputString`** in CamelCase, indem anhand von **`SplitChars`** festgelegt wird, welche(s) Zeichen die nächste Konvertierung in Großbuchstaben auslöst. **`SplitChars`** wird als Regular Expression verwendet, wenn **`IsRegex = true()`** oder als einfache Zeichen, wenn **`IsRegex = false()`**. Das erste Zeichen im Ausgabestring wird mit einem Großbuchstaben geschrieben.

Beispiele

- **`altova:camel-case`**`("setname getname", "set|get", true())` gibt `setName getName` zurück

- **altova:camel-case**("altova\documents\testcases", "\", false()) gibt Altova\Documents\Testcases zurück

▼ char [altova:]

altova:char(Position *as xs:integer*) als **xs:string** **XP3.1** **XQ3.1**
Gibt einen String zurück, der das Zeichen an der durch das Argument Position definierten Position enthält. Dieses Zeichen wird durch Konvertierung des Werts des Kontextelements in xs:string ermittelt. Der Ergebnisstring ist leer, wenn an dem durch das Position Argument gelieferten Index kein Zeichen vorhanden ist.

▣ *Beispiele*

Wenn das Kontextelement **1234ABCD** lautet:

- **altova:char**(2) gibt 2 zurück
- **altova:char**(5) gibt A zurück
- **altova:char**(9) gibt den leeren String zurück.
- **altova:char**(-2) gibt den leeren String zurück.

altova:char(InputString *als xs:string*, **Position** *als xs:integer*) als **xs:string** **XP3.1** **XQ3.1**
Gibt einen String zurück, der das Zeichen enthält, das sich in dem als InputString Argument gelieferten String an der durch das Argument Position definierten Position befindet. Der Ergebnisstring ist leer, wenn an dem durch das Position Argument gelieferten Index kein Zeichen vorhanden ist.

▣ *Beispiele*

- **altova:char**("2014-01-15", 5) gibt - zurück
- **altova:char**("USA", 1) gibt U zurück
- **altova:char**("USA", 1) gibt den leeren String zurück.
- **altova:char**("USA", -2) gibt den leeren String zurück.

▼ create-hash-from-string[altova:]

altova:create-hash-from-string(InputString *als xs:string*) als **xs:string** **XP2** **XQ1** **XP3.1** **XQ3.1**
altova:create-hash-from-string(InputString *als xs:string*, **HashAlgo** *als xs:string*) als **xs:string** **XP2** **XQ1** **XP3.1** **XQ3.1**
Generiert anhand von InputString mit Hilfe des durch das Argument HashAlgo definierten Hash-Algorithmus einen Hash-String. Es können die folgenden Hash-Algorithmen definiert werden (in Groß- oder Kleinbuchstaben): **MD5**, **SHA-1**, **SHA-224**, **SHA-256**, **SHA-384**, **SHA-512**. Wenn das zweite Argument nicht definiert ist (*siehe erste Signatur oben*), wird der Hash-Algorithmus **SHA-256** verwendet.

▣ *Beispiele*

- **altova:create-hash-from-string**('abc') gibt einen Hash-String zurück, der mit Hilfe des Hash-Algorithmus **SHA-256** generiert wurde.
- **altova:create-hash-from-string**('abc', 'md5') gibt einen Hash-String zurück, der mit Hilfe des Hash-Algorithmus **MD5** generiert wurde.
- **altova:create-hash-from-string**('abc', 'MD5') gibt einen Hash-String zurück, der mit Hilfe des Hash-Algorithmus **MD5** generiert wurde.

▼ first-chars [altova:]

altova:first-chars(X-Number *as xs:integer*) als **xs:string** **XP3.1** **XQ3.1**
Gibt einen String zurück, der die ersten x Zeichen (bezeichnet durch `X-Number`) des String enthält, der durch Konvertierung des Werts des Kontextelements in `xs:string` erzeugt wird.

Beispiele

Wenn das Kontextelement **1234ABCD** lautet:

- **altova:first-chars**(2) gibt 12 zurück
- **altova:first-chars**(5) gibt 1234A zurück
- **altova:first-chars**(9) gibt 1234ABCD zurück

altova:first-chars(InputString *als xs:string*, **X-Number** *als xs:integer*) als **xs:string** **XP3.1** **XQ3.1**
Gibt einen String zurück, der die ersten x Zeichen (bezeichnet durch `X-Number`) des String enthält, das als das Argument `InputString` angegeben ist.

Beispiele

- **altova:first-chars**("2014-01-15", 5) gibt 2014- zurück
- **altova:first-chars**("USA", 1) gibt U zurück

▼ format-string [altova:]

altova:format-string(InputString *als xs:string*, **FormatSequence** *als item()**) als **xs:string** **XP3.1** **XQ3.1**
Der Input String (erstes Argument) enthält Positionsparameter (`%1`, `%2`, usw.). Jeder Parameter wird durch das String-Element ersetzt, das sich in der (als zweites Argument bereitgestellten) Formatsequenz an der entsprechenden Position befindet. Daher ersetzt des erste Element in der Formatsequenz den Positionsparameter `%1`, das zweite den Positionsparameter `%2`, usw. Die Funktion gibt diesen formatierten String zurück, der die Ersetzungen enthält. Wenn für einen Positionsparameter kein String existiert, wird der Positionsparameter selbst zurückgegeben. Dies kommt vor, wenn der Index eines Positionsparameters größer als die Anzahl der Elemente in der Formatsequenz ist.

Beispiele

- **altova:format-string**('Hello %1, %2, %3', ('Jane','John','Joe')) gibt "Hello Jane, John, Joe" zurück.
- **altova:format-string**('Hello %1, %2, %3', ('Jane','John','Joe', 'Tom')) gibt "Hello Jane, John, Joe" zurück.
- **altova:format-string**('Hello %1, %2, %4', ('Jane','John','Joe', 'Tom')) gibt "Hello Jane, John, Tom" zurück.
- **altova:format-string**('Hello %1, %2, %4', ('Jane','John','Joe')) gibt "Hello Jane, John, %4" zurück.

▼ last-chars [altova:]

altova:last-chars(X-Number *als xs:integer*) als **xs:string** **XP3.1** **XQ3.1**
Gibt einen String zurück, der die letzten x Zeichen (bezeichnet durch `X-Number`) des String enthält, der durch Konvertierung des Werts des Kontextelements in `xs:string` erzeugt wird.

Beispiele

Wenn das Kontextelement `1234ABCD` lautet:

- `altova:last-chars(2)` gibt `CD` zurück
- `altova:last-chars(5)` gibt `4ABCD` zurück
- `altova:last-chars(9)` gibt `1234ABCD` zurück

altova:last-chars(InputString *als xs:string*, **X-Number** *als xs:integer*) als **xs:string** XP3.1 XQ3.1

Gibt einen String zurück, der die letzten x Zeichen (bezeichnet durch `X-Number`) des String enthält, das als das Argument `InputString` angegeben ist.

Beispiele

- `altova:last-chars("2014-01-15", 5)` gibt `01-15-` zurück
- `altova:last-chars("USA", 10)` gibt `USA` zurück

pad-string-left [altova:]

altova:pad-string-left(StringToPad *als xs:string*, **Repeats** *als xs:integer*, **PadCharacter** *als xs:string*) als **xs:string** XP3.1 XQ3.1

Das Argument `PadCharacter` ist ein einzelnes Zeichen. Es wird links vom String als Auffüllzeichen eingefügt, um die Anzahl der Zeichen in `StringToPad` zu erhöhen, damit diese Anzahl dem Ganzzahlwert des Arguments `StringLength` entspricht. Das Argument `StringLength` kann jeden beliebigen (positiven oder negativen) Ganzzahlwert haben, Auffüllzeichen werden aber nur verwendet, wenn der Wert von `StringLength` größer als die Anzahl der Zeichen in `StringToPad` ist. Wenn `StringToPad` mehr Zeichen als der Wert von `StringLength` hat, bleibt `StringToPad` unverändert.

Beispiele

- `altova:pad-string-left('AP', 1, 'Z')` gibt `'AP'` zurück
- `altova:pad-string-left('AP', 2, 'Z')` gibt `'AP'` zurück
- `altova:pad-string-left('AP', 3, 'Z')` gibt `'ZAP'` zurück
- `altova:pad-string-left('AP', 4, 'Z')` gibt `'ZZAP'` zurück
- `altova:pad-string-left('AP', -3, 'Z')` gibt `'AP'` zurück
- `altova:pad-string-left('AP', 3, 'YZ')`gibt einen Fehler zurück, dass das Auffüllzeichen zu lang ist

pad-string-right [altova:]

altova:pad-string-right(StringToPad *als xs:string*, **Repeats** *als xs:integer*, **PadCharacter** *als xs:string*) als **xs:string** XP3.1 XQ3.1

Das Argument `PadCharacter` ist ein einzelnes Zeichen. Es wird rechts vom String als Auffüllzeichen eingefügt, um die Anzahl der Zeichen in `StringToPad` zu erhöhen, damit diese Anzahl dem Ganzzahlwert des Arguments `StringLength` entspricht. Das Argument `StringLength` kann jeden beliebigen (positiven oder negativen) Ganzzahlwert haben, Auffüllzeichen werden aber nur verwendet, wenn der Wert von `StringLength` größer als die Anzahl der Zeichen in `StringToPad` ist. Wenn `StringToPad` mehr Zeichen als der Wert von `StringLength` hat, bleibt `StringToPad` unverändert.

Beispiele

- `altova:pad-string-right('AP', 1, 'Z')` gibt `'AP'` zurück
- `altova:pad-string-right('AP', 2, 'Z')` gibt `'AP'` zurück

- `altova:pad-string-right('AP', 3, 'Z')` gibt `'ZAP'` zurück
- `altova:pad-string-right('AP', 4, 'Z')` gibt `'ZZAP'` zurück
- `altova:pad-string-right('AP', -3, 'Z')` gibt `'AP'` zurück
- `altova:pad-string-right('AP', 3, 'YZ')` gibt einen Fehler zurück, dass das Auffüllzeichen zu lang ist

repeat-string [altova:]

altova:repeat-string(InputString *als xs:string*, **Repeats** *als xs:integer*) als **xs:string** **XP2** **XQ1** **XP3.1** **XQ3.1**

Generiert einen String, der sich zusammensetzt aus dem ersten `InputString`-Argument, das die Anzahl der `Repeats` wiederholt wird.

Beispiele

- `altova:repeat-string("Altova #", 3)` gibt `"Altova #Altova #Altova #"` zurück

substring-after-last [altova:]

altova:substring-after-last(MainString *als xs:string*, **CheckString** *als xs:string*) als **xs:string** **XP3.1** **XQ3.1**

Falls in `MainString` `CheckString` gefunden wird, so wird der Substring zurückgegeben, der in `MainString` nach `CheckString` steht. Falls `CheckString` in `MainString` nicht gefunden wird, so wird der leere String zurückgegeben. Wenn `CheckString` ein leerer String ist, so wird der gesamte `MainString` zurückgegeben. Falls `CheckString` mehrmals in `MainString`, vorkommt, so wird der Substring nach der letzten Instanz von `CheckString` zurückgegeben.

Beispiele

- `altova:substring-after-last('ABCDEFGH', 'B')` gibt `'CDEFGH'` zurück
- `altova:substring-after-last('ABCDEFGH', 'BC')` gibt `'DEFGH'`zurück
- `altova:substring-after-last('ABCDEFGH', 'BD')` gibt `''` zurück
- `altova:substring-after-last('ABCDEFGH', 'Z')` gibt `''` zurück
- `altova:substring-after-last('ABCDEFGH', '')` gibt `'ABCDEFGH'` zurück
- `altova:substring-after-last('ABCD-ABCD', 'B')` gibt `'CD'` zurück
- `altova:substring-after-last('ABCD-ABCD-ABCD', 'BCD')` gibt `''` zurück

substring-before-last [altova:]

altova:substring-before-last(MainString *as xs:string*, **CheckString** *as xs:string*) **as xs:string** **XP3.1** **XQ3.1**

Falls in `MainString` `CheckString` gefunden wird, so wird der Substring zurückgegeben, der in `MainString` vor `CheckString` steht. Falls `CheckString` in `MainString` nicht gefunden wird, so wird der leere String zurückgegeben. Wenn `CheckString` ein leerer String ist, so wird der gesamte `MainString` zurückgegeben. Falls `CheckString` mehrmals in `MainString`, vorkommt, so wird der Substring vor der letzten Instanz von `CheckString` zurückgegeben.

Beispiele

- `altova:substring-before-last('ABCDEFGH', 'B')` gibt `'A'` zurück
- `altova:substring-before-last('ABCDEFGH', 'BC')` gibt `'A'` zurück

- `altova:substring-before-last('ABCDEFGH', 'BD')` gibt `''` zurück
- `altova:substring-before-last('ABCDEFGH', 'Z')` gibt `''` zurück
- `altova:substring-before-last('ABCDEFGH', '')` gibt `''` zurück
- `altova:substring-before-last('ABCD-ABCD', 'B')` gibt `'ABCD-A'` zurück
- `altova:substring-before-last('ABCD-ABCD-ABCD', 'ABCD')` gibt `'ABCD-ABCD-'` zurück

substring-pos [altova:]

`altova:substring-pos(StringToCheck` *`als xs:string`*`, `**`StringToFind`** *`als xs:string`*`)` als **`xs:integer`** **XP3.1** **XQ3.1**

Gibt die Zeichenposition der ersten Instanz von `StringToFind` im String `StringToCheck` zurück. Die Zeichenposition wird in Form einer Ganzzahl angegeben. Das erste Zeichen von `StringToCheck` hat die Position `1`. Wenn `StringToFind` in `StringToCheck` nicht vorkommt, wird die Ganzzahl `0` zurückgegeben. Um den String auf eine zweite oder eine weiter hinten folgende Instanz von `StringToCheck` zu überprüfen, verwenden Sie die nächste Signatur dieser Funktion.

Beispiele

- `altova:substring-pos('Altova', 'to')` gibt `3` zurück
- `altova:substring-pos('Altova', 'tov')` gibt `3` zurück
- `altova:substring-pos('Altova', 'tv')` gibt `0` zurück
- `altova:substring-pos('AltovaAltova', 'to')` gibt `3` zurück

`altova:substring-pos(StringToCheck` *`als xs:string`*`, `**`StringToFind`** *`als xs:string`*`, `**`Integer`** *`als xs:integer`*`)` als **`xs:integer`** **XP3.1** **XQ3.1**

Gibt die Zeichenposition von `StringToFind` im String `StringToCheck` zurück. Die Suche nach `StringToFind` beginnt an der durch das Argument `Integer` angegebenen Zeichenposition; der Zeichen-Substring vor dieser Position wird nicht durchsucht. Die zurückgegebene Ganzzahl gibt jedoch die Position des gefundenen String innerhalb des *gesamten* String `StringToCheck` an. Diese Signatur dient dazu, die zweite oder eine weiter hinten folgende Position eines String zu finden, der mehrmals in `StringToCheck` vorkommt. Wenn `StringToFind` in `StringToCheck` nicht vorkommt, wird die Ganzzahl `0` zurückgegeben.

Beispiele

- `altova:substring-pos('Altova', 'to', 1)` gibt `3` zurück
- `altova:substring-pos('Altova', 'to', 3)` gibt `3` zurück
- `altova:substring-pos('Altova', 'to', 4)` gibt `0` zurück
- `altova:substring-pos('Altova-Altova', 'to', 0)` gibt `3` zurück
- `altova:substring-pos('Altova-Altova', 'to', 4)` gibt `10` zurück

trim-string [altova:]

`altova:trim-string(InputString` *`als xs:string`*`)` als **`xs:string`** **XP3.1** **XQ3.1**

Diese Funktion verwendet ein `xs:string` Argument, entfernt alle voran- und nachgestellten Leerzeichen und gibt einen "getrimmten" `xs:string` zurück.

Beispiele

- `altova:trim-string("  Hello World  ")` gibt `"Hello World"` zurück
- `altova:trim-string("Hello World  ")` gibt `"Hello World"` zurück
- `altova:trim-string("  Hello World")` gibt `"Hello World"` zurück

- **altova:trim-string**("Hello World") gibt "Hello World" zurück
- **altova:trim-string**("Hello World") gibt "Hello World" zurück

▼ trim-string-left [altova:]

altova:trim-string-left(InputString *als xs:string*) als **xs:string** **XP3.1** **XQ3.1**
Diese Funktion verwendet ein xs:string Argument, entfernt alle vorangestellten Leerzeichen und gibt einen "links getrimmten" xs:string zurück.

Beispiele

- **altova:trim-string-left**(" Hello World ") gibt "Hello World " zurück
- **altova:trim-string-left**("Hello World ") gibt "Hello World " zurück
- **altova:trim-string-left**(" Hello World") gibt "Hello World" zurück
- **altova:trim-string-left**("Hello World") gibt "Hello World" zurück
- **altova:trim-string-left**("Hello World") gibt "Hello World" zurück

▼ trim-string-right [altova:]

altova:trim-string-right(InputString *als xs:string*) als **xs:string** **XP3.1** **XQ3.1**
Diese Funktion verwendet ein xs:string Argument, entfernt alle nachgestellten Leerzeichen und gibt einen "rechts getrimmten" xs:string zurück.

Beispiele

- **altova:trim-string-right**(" Hello World ") gibt " Hello World" zurück
- **altova:trim-string-right**("Hello World ") gibt "Hello World" zurück
- **altova:trim-string-right**(" Hello World") gibt " Hello World" zurück
- **altova:trim-string-right**("Hello World") gibt "Hello World" zurück
- **altova:trim-string-right**("Hello World") gibt "Hello World" zurück

10.2.1.9 XPath/XQuery-Funktionen: Diverse Funktionen

Die folgenden XPath/XQuery-Funktionen für allgemeine Zwecke werden in der aktuellen Version von RaptorXML Server unterstützt und können in (i) in einem XSLT-Kontext in XPath-Ausdrücken oder (ii) in einem XQuery-Dokument in XQuery-Ausdrücken verwendet werden.

Anmerkung zur Benennung von Funktionen und zur Anwendbarkeit der Sprache

Altova-Erweiterungsfunktionen können in XPath/XQuery-Ausdrücken verwendet werden. Dadurch stehen neben den Funktionen in der Standardbibliothek der XPath-, XQuery- und XSLT-Funktionen zusätzliche Funktionen zur Verfügung. Die Altova-Erweiterungsfunktionen befinden sich im **Altova-Erweiterungsfunktions-Namespace, http://www.altova.com/xslt-extensions** und sind in diesem Abschnitt mit dem Präfix **altova:**, das als an diesen Namespace gebunden angenommen wird, gekennzeichnet. Beachten Sie, dass manche Funktionen in zukünftigen Versionen Ihres Produkts eventuell nicht mehr unterstützt werden oder dass sich das Verhalten einzelner Funktionen ändern kann. Um zu sehen, welche Altova-Erweiterungsfunktionen unterstützt werden, lesen Sie bitte die Dokumentation zur jeweiligen Release.

XPath-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	XP1 XP2 XP3.1
XSLT-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	XSLT1 XSLT2 XSLT3
XQuery-Funktionen (in XQuery-Ausdrücken in XQuery verwendet):	XQ1 XQ3.1

decode-string [altova:]

altova:decode-string(Input *als xs:base64Binary*) **als xs:string** XP3.1 XQ3.1
altova:decode-string(Input *als xs:base64Binary*, **Encoding** *als xs:string*) **als xs:string** XP3.1 XQ3.1
Dekodiert den angegebenen base64Binary-Input anhand der definierten Kodierung zu einem String. Wenn keine Kodierung definiert ist, wird die UTF-8-Kodierung verwendet. Die folgenden Kodierungen werden unterstützt: `US-ASCII, ISO-8859-1, UTF-16, UTF-16LE, UTF-16BE, ISO-10646-UCS2, UTF-32, UTF-32LE, UTF-32BE, ISO-10646-UCS4`

Beispiele

- `altova:decode-string($XML1/MailData/Meta/b64B)` gibt den base64Binary-Input als UTF-8-kodierten String zurück.
- `altova:decode-string($XML1/MailData/Meta/b64B, "UTF-8")` gibt den base64Binary-Input als UTF-8-kodierten String zurück.
- `altova:decode-string($XML1/MailData/Meta/b64B, "ISO-8859-1")` gibt den base64Binary-Input als ISO-8859-1-kodierten String zurück.

encode-string [altova:]

altova:encode-string(InputString *als xs:string*) **als xs:base64Binaryinteger** XP3.1 XQ3.1
altova:encode-string(InputString *als xs:string*, **Encoding** *als xs:string*) **als xs:base64Binaryinteger** XP3.1 XQ3.1
Kodiert den angegebenen String gemäß der definierten Kodierung, falls eine angegeben wird. Wenn keine Kodierung definiert ist, wird die UTF-8-Kodierung verwendet. Der kodierte String wird in base64Binary-Zeichen konvertiert und es wird der konvertierte base64Binary-Wert zurückgegeben. Anfangs wird die UTF-8-Kodierung unterstützt. Die Unterstützung wird auf die folgenden Kodierungen ausgeweitet werden: `US-ASCII, ISO-8859-1, UTF-16, UTF-16LE, UTF-16BE, ISO-10646-UCS2, UTF-32, UTF-32LE, UTF-32BE, ISO-10646-UCS4`

Beispiele

- `altova:encode-string("Altova")` gibt das base64Binary-Äquivalent des UTF-8-kodierten String `"Altova"` zurück.
- `altova:encode-string("Altova", "UTF-8")` gibt das base64Binary-Äquivalent des UTF-8-kodierten String `"Altova"` zurück.

get-temp-folder [altova:]

altova:get-temp-folder() als `xs:string` XP2 XQ1 XP3.1 XQ3.1
Diese Funktion hat kein Argument. Sie gibt den Pfad zum temporären Ordner des aktuellen Benutzers zurück.

Beispiele

- **altova:get-temp-folder**() würde auf einem Windows-Rechner z.B. den folgenden Pfad als xs:string zurückgeben: C:\Users\<UserName>\AppData\Local\Temp\.

generate-guid [altova:]

altova:generate-guid() als **xs:string** **XP2** **XQ1** **XP3.1** **XQ3.1**
Generiert einen eindeutigen String GUID-String.

Beispiele

- **altova:generate-guid**() gibt (z.B.) 85F971DA-17F3-4E4E-994E-99137873ACCD zurück

high-res-timer [altova:]

altova:high-res-timer() als **xs:double** **XP3.1** **XQ3.1**
Gibt einen hochauflösenden System-Timer-Wert in Sekunden zurück. Wenn in einem System ein hochauflösender Timer zur Verfügung steht, können bei Bedarf (z.B. bei Animationen und zur Ermittlung des exakten Codeausführungszeitpunkts) hochauflösende Zeitmessungen vorgenommen werden. Diese Funktion stellt die Auflösung des Hochauflösungs-Timers des Systems zur Verfügung.

Beispiele

- **altova:high-res-timer**() gibt eine Wert wie '1.16766146154566E6' zurück.

parse-html [altova:]

altova:parse-html(HTMLText *als xs:string***)** als **node()** **XP3.1** **XQ3.1**
Das Argument HTMLText ist ein String, der den Text eines HTML-Dokuments enthält. Die Funktion erstellt anhand des Strings eine HTML-Struktur. Der bereitgestellte String kann das HTML-Element enthalten, muss dies aber nicht tun. In beiden Fällen ist das Root-Element der Struktur ein Element namens **HTML**. Sie sollten sicher stellen, dass der HTML-Code im bereitgestellten String gültiger HTML-Code ist.

Beispiel

- **altova:parse-html**("<html><head/><body><h1>Header</h1></body></html>") erstellt anhand des bereitgestellten Strings eine HTML-Struktur.

sleep[altova:]

altova:sleep(Millisecs *als xs:integer***)** als **empty-sequence()** **XP2** **XQ1** **XP3.1** **XQ3.1**
Unterbricht die Ausführung der aktuellen Operation für die Anzahl der durch das Argument Millisecs angegebenen Millisekunden.

Beispiel

- **altova:sleep**(1000) unterbricht die Ausführung der aktuellen Operation für 1000 Millisekunden.

[Nach oben 511]

10.2.1.10 Diagrammfunktionen

Mit Hilfe der unten aufgelisteten Diagrammfunktionen können Sie Diagramme als Bilder erstellen, generieren und speichern. Sie werden in der aktuellen Version Ihres Altova-Produkts auf die unten beschriebene Art unterstützt. Beachten Sie jedoch, dass eine oder mehrere dieser Funktionen in zukünftigen Produktversionen eventuell nicht mehr unterstützt werden, bzw. dass sich das Verhalten einzelner Funktionen ändern kann. Um Informationen über die Unterstützung für Altova Erweiterungsfunktionen in der jeweiligen Release zu erhalten, schlagen Sie bitte in der Dokumentation der jeweils aktuellen Release nach.

Anmerkung: Diagrammfunktionen werden nur in **Altova Server-Produkten** und den **Enterprise-Editionen von Altova-Produkten** unterstützt.

Anmerkung: Unterstützte Bildformate für Diagramme in Server-Editionen sind `jpg`, `png` und `bmp`. Die beste Option ist png, da dabei keine Daten verloren gehen und es sich um ein komprimiertes Format handelt. In Enterprise-Editionen werden die folgenden Formate unterstützt: `jpg`. `png`, `bmp` und `gif`.

Funktionen zum Generieren und Speichern von Diagrammen

Diese Funktionen generieren anhand des (mit Hilfe der Diagrammerstellungsfunktionen) erzeugten Diagrammobjekts entweder ein Bild oder speichern ein Bild in einer Datei

`altova:generate-chart-image` `($chart, $width, $height, $encoding) als atomic`

wobei

- `$chart` das Diagramm-Erweiterungsobjekt ist, das mit der Funktion `altova:create-chart` erzeugt wurde
- `$width` und `$height` mit einer Längeneinheit definiert werden müssen
- `$encoding` kann `x-binarytobase64` oder `x-binarytobase16` sein

Die Funktion gibt das Diagrammbild in der definierten Kodierung zurück.

`altova:generate-chart-image` `($chart, $width, $height, $encoding, $imagetype) als atomic`

wobei

- `$chart` das Diagramm-Erweiterungsobjekt ist, das mit der Funktion `altova:create-chart` erzeugt wurde
- `$width` und `$height` mit einer Längeneinheit definiert werden müssen
- `$encoding` kann `x-binarytobase64` oder `x-binarytobase16` sein
- `$imagetype` eines der folgenden Bildformate sein kann: `png`, `gif`, `bmp`, `jpg`, `jpeg`. Beachten Sie, dass `gif` in Server-Produkten nicht unterstützt wird. *Siehe oben auf dieser Seite*.

Die Funktion gibt das Diagrammbild in der definierten Kodierung und im definierten Bildformat zurück.

`altova:save-chart-image` `($chart, $filename, $width, $height) als empty()` ***(nur Windows)***

wobei

- `$chart` das Diagramm-Erweiterungsobjekt ist, das mit der Funktion `altova:create-chart` erzeugt wurde
- `$filename` der Pfad und Name der Datei ist, unter dem das Diagrammbild gespeichert werden soll
- `$width` und `$height` mit einer Längeneinheit definiert werden müssen

Die Funktion speichert das Diagrammbild unter dem in `$filename` definierten Dateinamen. Alternativ zu dieser Funktion könnten Sie auch die Funktion `xsl:result-document` mit `encoding="x-base64tobinary"` verwenden, wobei der image-data-Inhalt entweder über die Funktion `generate-chart-image()` oder die Funktion `chart()` erhalten wird.

`altova:save-chart-image` `($chart, $filename, $width, $height, $imagetype) als empty()` ***(nur Windows)***

wobei

- `$chart` das Diagramm-Erweiterungsobjekt ist, das mit der Funktion `altova:create-chart` erzeugt wurde
- `$filename` der Pfad und Name der Datei ist, unter dem das Diagrammbild gespeichert werden soll
- `$width` und `$height` mit einer Längeneinheit definiert werden müssen
- `$imagetype` eines der folgenden Bildformate sein kann: `png`, `gif`, `bmp`, `jpg`, `jpeg`. Beachten Sie, dass `gif` in Server-Produkten nicht unterstützt wird. *Siehe oben auf dieser Seite*.

Die Funktion speichert das Diagrammbild im definierten Bildformat unter dem in `$filename` definierten Dateinamen. Alternativ zu dieser Funktion könnten Sie auch die Funktion `xsl:result-document` mit `encoding="x-base64tobinary"` verwenden, wobei der image-data-Inhalt entweder über die Funktion `generate-chart-image()` oder die Funktion `chart()` erhalten wird.

Funktionen zur Erstellung von Diagrammen

Die folgenden Funktionen dienen zur Erstellung von Diagrammen.

`altova:create-chart``($chart-config, $chart-data-series*) als chart extension item`

wobei

- `$chart-config` das Diagrammkonfigurations-Erweiterungsobjekt ist, das mit der Funktion `altova:create-chart-config` oder über die Funktion `altova:create-chart-config-from-xml` erzeugt wurde
- `$chart-data-series` das chart-data-series-Erweiterungsobjekt ist, das mit der Funktion `altova:create-chart-data-series` oder der Funktion `altova:create-chart-data-series-from-rows` abgerufen wird

Die Funktion gibt ein Diagrammerweiterungsobjekt zurück, das anhand der über die Argumente gelieferten Daten erzeugt wird.

`altova:chart``($chart-config, $chart-data-series*) als chart extension item`

wobei

- `$chart-config` das Diagrammkonfigurations-Erweiterungsobjekt ist. Es handelt sich hierbei um eine nicht geordnete Reihe von vier Schlüssel:Wert-Paaren, wobei die vier Schlüssel `"width"`, `"height"`, `"title"` und `"kind"` sind. Die Werte von `width` und `height` sind Ganzzahlen und definieren die Höhe und Breite des Diagramms in Pixel. Der Wert von `kind` ist einer der folgenden: `Pie, Pie3d, BarChart, BarChart3d, BarChart3dGrouped, LineChart, ValueLineChart, RoundGauge, BarGauge`.
- `$chart-data-series` ist jeweils ein Array der Größe 3, wobei jedes Array eine Diagrammdatenreihe definiert. Jedes Array besteht aus: (i) dem Namen der Datenreihe, (ii) den Werten der x-Achse, (iii) den Werten der y-Achse. Es können mehrere Datenreihen angegeben werden. Im Beispiel unten enthalten die zwei Arrays Daten für die Temperatur-Monatsminima bzw. -maxima.

Die Funktion gibt ein Element vom Typ `xs:base64Binary` zurück, welches das Diagrammbild enthält. Dieses Bild wird anhand der über die Argumente der Funktion bereitgestellten Daten erstellt. Beachten Sie, dass diese Funktion, da sie Arrays und Zuordnungen verwendet, nur in XPath 3.1, XQuery 3.1 oder XSLT 3.0 verwendet werden kann.

Beispiel:
```
altova:chart( map{'width':800, 'height':600, "kind":"LineChart", "title":"Monthly Temperatures"}, (['Min', $temps/Month, $temps/Month/@min], ['Max', $temps/Month, $temps/Month/@max]) )
```

`altova:create-chart-config``($type-name, $title) als chart-config Erweiterungsobjekt`

wobei

- `$type-name` den Typ des zu erstellenden Diagramms definiert: `Pie, Pie3d, BarChart, BarChart3d, BarChart3dGrouped, LineChart, ValueLineChart, RoundGauge, BarGauge`
- `$title` der Name des Diagramms ist

Die Funktion gibt ein Diagrammkonfigurations-Erweiterungsobjekt zurück, das die Konfigurationsinformationen zum Diagramm enthält.

`altova:create-chart-config-from-xml``($xml-struct) als chart-config Erweiterungsobjekt`

wobei

- `$xml-struct` die XML-Struktur ist, die die Konfigurationsinformationen des Diagramms enthält

Die Funktion gibt ein Diagrammkonfigurations-Erweiterungsobjekt zurück, das die Konfigurationsinformationen zum Diagramm enthält. Diese Informationen werden in einem XML-Datenfragment[518] geliefert.

`altova:create-chart-data-series``($series-name?, $x-values*, $y-values*) als chart-data-series Erweiterungsobjekt`

wobei

- `$series-name` der Name der Datenreihe ist
- `$x-values` die Liste der Werte für die X-Achse liefert
- `$y-values` die Liste der Werte für die Y-Achse liefert

Die Funktion gibt ein Diagrammdatenreihen-Erweiterungsobjekt zurück, das die Daten zur Erstellung des Diagramms, also die Namen der Datenreihen, und die Achsendaten enthält.

altova:create-chart-data-row(x, y1, y2, y3, ...) als chart-data-x-*N*y-row Erweiterungsobjekt

wobei

- `x` der Wert der X-Achsen-Spalte der Diagrammdatenzeile ist
- `yN` die Werte der Spalten für die Y-Achse sind

Die Funktion gibt ein chart-data-x-Ny-row Erweiterungsobjekt zurück, das die Daten für die X-Achsen-Spalte und die Y-Achsen-Spalten einer einzigen Datenreihe enthält.

altova:create-chart-data-series-from-rows($series-names as xs:string*, $row*) als chart-data-series-Erweiterungsobjekt

wobei

- `$series-name` der Name der zu erstellenden Datenreihen ist
- `$row das` chart-data-x-Ny-row Erweiterungsobjekt ist, das als Datenreihe erstellt werden soll

Die Funktion gibt ein chart-data-series Erweiterungsobjekt zurück, das die Daten für die X- und die Y-Achse der Datenreihe enthält.

altova:create-chart-layer($chart-config, $chart-data-series*) als chart-layer Erweiterungsobjekt

wobei

- `$chart-config` das Diagrammkonfigurations-Erweiterungsobjekt ist, das mit der Funktion `altova:create-chart-config` oder über die Funktion `altova:create-chart-config-from-xml` erzeugt wurde
- `$chart-data-series` das chart-data-series-Erweiterungsobjekt ist, das mit der Funktion `altova:create-chart-data-series` oder der Funktion `altova:create-chart-data-series-from-rows` abgerufen wird

Die Funktion gibt ein chart-layer Erweiterungsobjekt zurück, das chart-layer-Daten enthält.

altova:create-multi-layer-chart($chart-config, $chart-data-series*, $chart-layer*)

wobei

- `$chart-config` das Diagrammkonfigurations-Erweiterungsobjekt ist, das mit der Funktion `altova:create-chart-config` oder über die Funktion `altova:create-chart-config-from-xml` erzeugt wurde
- `$chart-data-series` das chart-data-series-Erweiterungsobjekt ist, das mit der Funktion `altova:create-chart-data-series` oder der Funktion `altova:create-chart-data-series-from-rows` abgerufen wird

- `$chart-layer` das chart-layer-Erweiterungsobjekt ist, das mit der Funktion `altova:create-chart-layer` abgerufen wird

Die Funktion gibt ein multi-layer-chart-Objekt zurück.

`altova:create-multi-layer-chart``($chart-config, $chart-data-series*, $chart-layer*, xs:boolean $mergecategoryvalues)`

wobei

- `$chart-config` das Diagrammkonfigurations-Erweiterungsobjekt ist, das mit der Funktion `altova:create-chart-config` oder über die Funktion `altova:create-chart-config-from-xml` erzeugt wurde
- `$chart-data-series` das chart-data-series-Erweiterungsobjekt ist, das mit der Funktion `altova:create-chart-data-series` oder der Funktion `altova:create-chart-data-series-from-rows` abgerufen wird
- `$chart-layer` das chart-layer-Erweiterungsobjekt ist, das mit der Funktion `altova:create-chart-layer` abgerufen wird
- `$mergecategoryvalues` bei `true` die Werte mehrerer Datenreihen zusammenführt und dies bei `false` nicht tut.

Die Funktion gibt ein multi-layer-chart-Objekt zurück.

10.2.1.10.1 XML-Struktur von Diagrammdaten

Unten sehen Sie die XML-Struktur von Diagrammdaten, wie sie für Altova-Erweiterungsfunktionen für Diagramme 514 angezeigt werden könnte. Diese Funktionen beeinflussen das Aussehen der einzelnen Diagramme. Nicht alle Elemente werden für alle Diagrammarten verwendet, so wird z.B. das Element <Pie> bei Balkendiagrammen ignoriert.

Anmerkung: Diagrammfunktionen werden nur in **Enterprise und Server-Editionen** von Altova Produkten unterstützt.

```
<chart-config>
        <General
        SettingsVersion="1"   must be provided
        ChartKind="BarChart"   Pie, Pie3d, BarChart, StackedBarChart, BarChart3d, BarChart3dGrouped,
LineChart, ValueLineChart, AreaChart, StackedAreaChart, RoundGauge, BarGauge, CandleStick
        BKColor="#ffffff"   Color
        BKColorGradientEnd="#ffffff"   Color. In case of a gradient, BKColor and BKColorGradientEnd
define the gradient's colors
        BKMode="#ffffff"   Solid, HorzGradient, VertGradient
        BKFile="Path+Filename"   String. If file exists, its content is drawn over the background.
        BKFileMode="Stretch"   Stretch, ZoomToFit, Center, Tile
        ShowBorder="1"   Bool
        PlotBorderColor="#000000"   Color
        PlotBKColor="#ffffff"   Color
        Title=""   String
        ShowLegend="1"   Bool
```



```
 OutsideMargin="3.%"   PercentOrPixel
 TitleToPlotMargin="3.%"   PercentOrPixel
 LegendToPlotMargin="3.%"   PercentOrPixel
 Orientation="vert"   Enumeration: possible values are: vert, horz
 >
 <TitleFont
    Color="#000000"   Color
    Name="Tahoma"   String
    Bold="1"   Bool
    Italic="0"   Bool
    Underline="0"   Bool
    MinFontHeight="10.pt"   FontSize (only pt values)
    Size="8.%"   FontSize />
 <LegendFont
    Color="#000000"
    Name="Tahoma"
    Bold="0"
    Italic="0"
    Underline="0"
    MinFontHeight="10.pt"
    Size="3.5%" />
 <AxisLabelFont
    Color="#000000"
    Name="Tahoma"
    Bold="1"
    Italic="0"
    Underline="0"
    MinFontHeight="10.pt"
    Size="5.%" />
 </General>

  <Line
ConnectionShapeSize="1.%"   PercentOrPixel
DrawFilledConnectionShapes="1"   Bool
DrawOutlineConnectionShapes="0"   Bool
DrawSlashConnectionShapes="0"   Bool
DrawBackslashConnectionShapes="0"   Bool
  />

  <Bar
ShowShadow="1"   Bool
ShadowColor="#a0a0a0"   Color
OutlineColor="#000000"   Color
ShowOutline="1"   Bool
  />

  <Area
Transparency="0"   UINT ( 0-255 ) 255 is fully transparent, 0 is opaque
OutlineColor="#000000"   Color
ShowOutline="1"   Bool
  />

  <CandleStick
```

```
      FillHighClose="0"   Bool. If 0, the body is left empty. If 1, FillColorHighClose is used for the candle
body
      FillColorHighClose="#ffffff"   Color. For the candle body when close > open
      FillHighOpenWithSeriesColor="1"   Bool. If true, the series color is used to fill the candlebody when
open > close
      FillColorHighOpen="#000000"   Color. For the candle body when open > close and
FillHighOpenWithSeriesColor is false
       />

   <Colors   User-defined color scheme: By default this element is empty except for the style and has no
Color attributes
      UseSubsequentColors ="1"  Boolean. If 0, then color in overlay is used. If 1, then subsequent colors
from previous chart layer is used
      Style="User"   Possible values are: "Default", "Grayscale", "Colorful", "Pastel", "User"
      Colors="#52aca0"   Color: only added for user defined color set
      Colors1="#d3c15d"   Color: only added for user defined color set
      Colors2="#8971d8"   Color: only added for user defined color set
      ...
      ColorsN=""   Up to ten colors are allowed in a set: from Colors to Colors9
       </Colors>

       <Pie
      ShowLabels="1"   Bool
      OutlineColor="#404040"   Color
      ShowOutline="1"   Bool
      StartAngle="0."   Double
      Clockwise="1"   Bool
      Draw2dHighlights="1"   Bool
      Transparency="0"   Int (0 to 255: 0 is opaque, 255 is fully transparent)
      DropShadowColor="#c0c0c0"   Color
      DropShadowSize="5.%"   PercentOrPixel
      PieHeight="10.%"   PercentOrPixel. Pixel values might be different in the result because of 3d tilting
      Tilt="40.0"   Double (10 to 90: The 3d tilt in degrees of a 3d pie)
      ShowDropShadow="1"   Bool
      ChartToLabelMargin="10.%"   PercentOrPixel
      AddValueToLabel="0"   Bool
      AddPercentToLabel="0"   Bool
      AddPercentToLabels_DecimalDigits="0"  UINT ( 0 – 2 )
      >
      <LabelFont
         Color="#000000"
         Name="Arial"
         Bold="0"
         Italic="0"
         Underline="0"
         MinFontHeight="10.pt"
         Size="4.%" />
       </Pie>

       <XY>
      <XAxis   Axis
         AutoRange="1"   Bool
         AutoRangeIncludesZero="1"   Bool
```

```
    RangeFrom="0."   Double: manual range
    RangeTill="1."   Double : manual range
    LabelToAxisMargin="3.%"   PercentOrPixel
    AxisLabel=""   String
    AxisColor="#000000"   Color
    AxisGridColor="#e6e6e6"   Color
    ShowGrid="1"   Bool
    UseAutoTick="1"   Bool
    ManualTickInterval="1."   Double
    AxisToChartMargin="0.px"   PercentOrPixel
    TickSize="3.px"   PercentOrPixel
    ShowTicks="1"   Bool
    ShowValues="1"   Bool
    AxisPosition="LeftOrBottom"   Enums: "LeftOrBottom", "RightOrTop", "AtValue"
    AxisPositionAtValue = "0"   Double
    >
    <ValueFont
       Color="#000000"
       Name="Tahoma"
       Bold="0"
       Italic="0"
       Underline="0"
       MinFontHeight="10.pt"
       Size="3.%"      />
</XAxis>
<YAxis   Axis (same as for XAxis)
   AutoRange="1"
   AutoRangeIncludesZero="1"
   RangeFrom="0."
   RangeTill="1."
   LabelToAxisMargin="3.%"
   AxisLabel=""
   AxisColor="#000000"
   AxisGridColor="#e6e6e6"
   ShowGrid="1"
   UseAutoTick="1"
   ManualTickInterval="1."
   AxisToChartMargin="0.px"
   TickSize="3.px"
   ShowTicks="1"   Bool
   ShowValues="1"   Bool
   AxisPosition="LeftOrBottom"   Enums: "LeftOrBottom", "RightOrTop", "AtValue"
   AxisPositionAtValue = "0"   Double
   >
   <ValueFont
      Color="#000000"
      Name="Tahoma"
      Bold="0"
      Italic="0"
      Underline="0"
      MinFontHeight="10.pt"
      Size="3.%"/>
</YAxis>
  </XY>
```

<XY3d

`AxisAutoSize="1"` *Bool: If false, XSize and YSize define the aspect ration of x and y axis. If true, aspect ratio is equal to chart window*

`XSize="100.%"` *PercentOrPixel. Pixel values might be different in the result because of 3d tilting and zooming to fit chart*

`YSize="100.%"` *PercentOrPixel. Pixel values might be different in the result because of 3d tilting and zooming to fit chart*

`SeriesMargin="30.%"` *PercentOrPixel. Pixel values might be different in the result because of 3d tilting and zooming to fit chart*

`Tilt="20."` *Double. -90 to +90 degrees*

`Rot="20."` *Double. -359 to +359 degrees*

`FoV="50.">` *Double. Field of view: 1-120 degree*

```
>
<ZAxis
   AutoRange="1"
   AutoRangeIncludesZero="1"
   RangeFrom="0."
   RangeTill="1."
   LabelToAxisMargin="3.%"
   AxisLabel=""
   AxisColor="#000000"
   AxisGridColor="#e6e6e6"
   ShowGrid="1"
   UseAutoTick="1"
   ManualTickInterval="1."
   AxisToChartMargin="0.px"
   TickSize="3.px" >
   <ValueFont
      Color="#000000"
      Name="Tahoma"
      Bold="0"
      Italic="0"
      Underline="0"
      MinFontHeight="10.pt"
      Size="3.%"/>
</ZAxis>
```

</XY3d>

<Gauge

`MinVal="0."` *Double*

`MaxVal="100."` *Double*

`MinAngle="225"` *UINT: -359-359*

`SweepAngle="270"` *UINT: 1-359*

`BorderToTick="1.%"` *PercentOrPixel*

`MajorTickWidth="3.px"` *PercentOrPixel*

`MajorTickLength="4.%"` *PercentOrPixel*

`MinorTickWidth="1.px"` *PercentOrPixel*

`MinorTickLength="3.%"` *PercentOrPixel*

`BorderColor="#a0a0a0"` *Color*

`FillColor="#303535"` *Color*

`MajorTickColor="#a0c0b0"` *Color*

`MinorTickColor="#a0c0b0"` *Color*

`BorderWidth="2.%"` *PercentOrPixel*

`NeedleBaseWidth="1.5%"` *PercentOrPixel*


```
      NeedleBaseRadius="5.%"   PercentOrPixel
      NeedleColor="#f00000"   Color
      NeedleBaseColor="#141414"   Color
      TickToTickValueMargin="5.%"   PercentOrPixel
      MajorTickStep="10."   Double
      MinorTickStep="5."   Double
      RoundGaugeBorderToColorRange="0.%"   PercentOrPixel
      RoundGaugeColorRangeWidth ="6.%"   PercentOrPixel
      BarGaugeRadius="5.%"   PercentOrPixel
      BarGaugeMaxHeight="20.%"   PercentOrPixel
      RoundGaugeNeedleLength="45.%"   PercentOrPixel
      BarGaugeNeedleLength="3.%"   PercentOrPixel
      >
      <TicksFont
         Color="#a0c0b0"
         Name="Tahoma"
         Bold="0"
         Italic="0"
         Underline="0"
         MinFontHeight="10.pt"
         Size="4.%"
      />
      <ColorRanges>   User-defined color ranges. By default empty with no child element entries
         <Entry
            From="50. "   Double
            FillWithColor="1"   Bool
            Color="#00ff00"   Color
         />
         <Entry
            From="50.0"
            FillWithColor="1"
            Color="#ff0000"
         />
         ...
      </ColorRanges>
       </Gauge>
</chart-config>
```

10.2.1.10.2 Beispiel: Diagrammfunktionen

Anhand des XSLT-Beispieldokuments weiter unten sehen Sie, wie Altova-Erweiterungsfunktionen für Diagramme 514 eingesetzt werden können. Weiter unten sehen Sie ein XML-Dokument und eine Abbildung des Ausgabebilds, das generiert wird, wenn ein XML-Dokument mit dem XSLT 2.0- oder XSLT 3.0-Prozessor anhand des XSLT-Dokuments verarbeitet wird.

Anmerkung: Diagrammfunktionen werden nur in **Enterprise und Server-Editionen** von Altova Produkten unterstützt.

Anmerkung: Weitere Informationen zur Erstellung von Diagrammdatentabellen finden Sie in der Dokumentation zu den Altova-Produkten XMLSpy und StyleVision.

XSLT-Dokument

In diesem (*unten aufgelisteten*) XSLT-Dokument wird mit Hilfe der Altova Diagramm-Erweiterungsfunktionen ein Kreisdiagramm generiert. Das XSLT-Dokument kann zur Verarbeitung des weiter unten angeführten XML-Dokuments verwendet werden.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:altovaext="http://www.altova.com/xslt-extensions"
    exclude-result-prefixes="#all">
    <xsl:output version="4.0" method="html" indent="yes" encoding="UTF-8"/>
    <xsl:template match="/">
        <html>
            <head>
                <title>
                    <xsl:text>HTML Page with Embedded Chart</xsl:text>
                </title>
            </head>
            <body>
                <xsl:for-each select="/Data/Region[1]">
                    <xsl:variable name="extChartConfig" as="item()*">
                        <xsl:variable name="ext-chart-settings" as="item()*">
                            <chart-config>
                                <General
                                    SettingsVersion="1"
                                    ChartKind="Pie3d"
                                    BKColor="#ffffff"
                                    ShowBorder="1"
                                    PlotBorderColor="#000000"
                                    PlotBKColor="#ffffff"
                                    Title="{@id}"
                                    ShowLegend="1"
                                    OutsideMargin="3.2%"
                                    TitleToPlotMargin="3.%"
                                    LegendToPlotMargin="6.%"
                                    >
                                    <TitleFont
                                        Color="#023d7d"
                                        Name="Tahoma"
                                        Bold="1"
                                        Italic="0"
                                        Underline="0"
                                        MinFontHeight="10.pt"
                                        Size="8.%" />
                                </General>
                            </chart-config>
                        </xsl:variable>
                        <xsl:sequence select="altovaext:create-chart-config-from-xml( $ext-
chart-settings )"/>
                    </xsl:variable>
```

```
                <xsl:variable name="chartDataSeries" as="item()*">
                    <xsl:variable name="chartDataRows" as="item()*">
                        <xsl:for-each select="(Year)">
                            <xsl:sequence select="altovaext:create-chart-data-row( (@id),
( .) )"/>
                        </xsl:for-each>
                    </xsl:variable>
                    <xsl:variable name="chartDataSeriesNames" as="xs:string*"
select=" ( (&quot;Series 1&quot;), &apos;&apos; )[1]"/>
                    <xsl:sequence
                        select="altovaext:create-chart-data-series-from-
rows( $chartDataSeriesNames, $chartDataRows)"/>
                </xsl:variable>
                <xsl:variable name="ChartObj" select="altovaext:create-
chart( $extChartConfig, ( $chartDataSeries), false() )"/>
                <xsl:variable name="sChartFileName" select="'mychart1.png'"/>
                <img src="{$sChartFileName, altovaext:save-chart-image( $ChartObj,
$sChartFileName, 400, 400 ) }"/>
            </xsl:for-each>
        </body>
    </html>
</xsl:template>
</xsl:stylesheet>
```

XML-Dokument

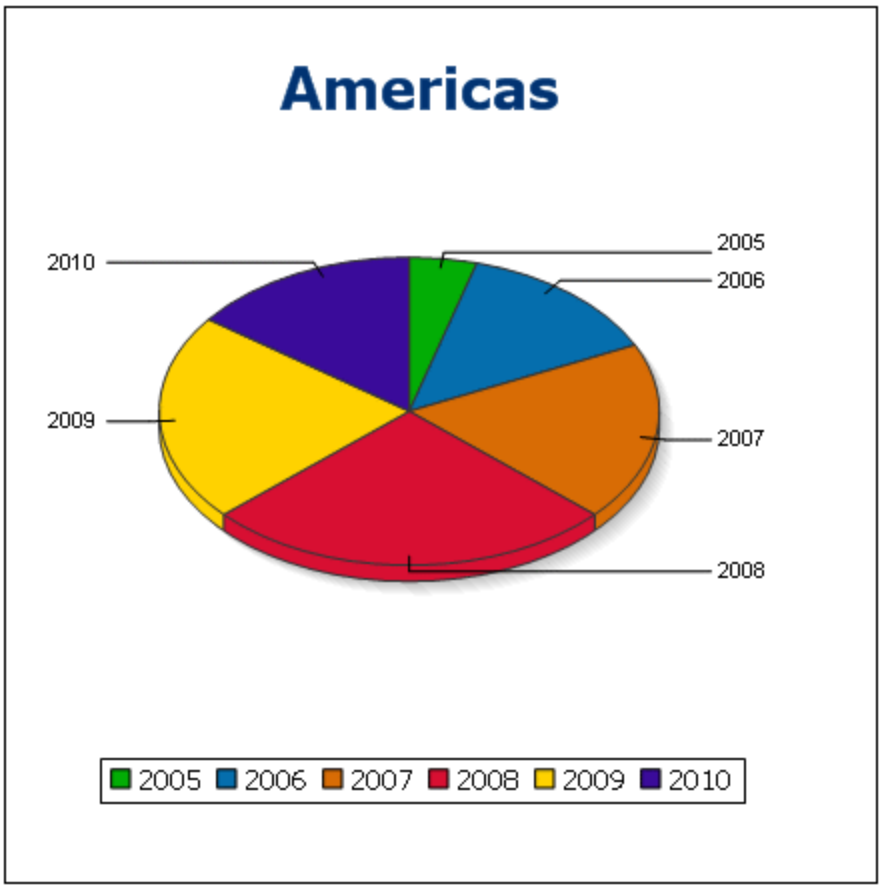
Dieses XML-Dokument kann mit dem oben stehenden XSLT-Dokument verarbeitet werden. Anhand der Daten im XML-Dokument wird das in der unten stehenden Abbildung gezeigte Kreisdiagramm generiert.

```
<?xml version="1.0" encoding="UTF-8"?>
<Data xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="YearlySales.xsd">
   <ChartType>Pie Chart 2D</ChartType>
   <Region id="Americas">
      <Year id="2005">30000</Year>
      <Year id="2006">90000</Year>
      <Year id="2007">120000</Year>
      <Year id="2008">180000</Year>
      <Year id="2009">140000</Year>
      <Year id="2010">100000</Year>
   </Region>
   <Region id="Europe">
      <Year id="2005">50000</Year>
      <Year id="2006">60000</Year>
      <Year id="2007">80000</Year>
      <Year id="2008">100000</Year>
      <Year id="2009">95000</Year>
      <Year id="2010">80000</Year>
   </Region>
   <Region id="Asia">
      <Year id="2005">10000</Year>
      <Year id="2006">25000</Year>
```

```
        <Year id="2007">70000</Year>
        <Year id="2008">110000</Year>
        <Year id="2009">125000</Year>
        <Year id="2010">150000</Year>
    </Region>
</Data>
```

Ausgabebild

Das unten gezeigt Kreisdiagramm wird generiert, wenn das oben aufgelistete XML-Dokument mit Hilfe des XSLT-Dokuments verarbeitet wird.



10.2.1.11 Barcode-Funktionen

Der XSLT-Prozessor verwendet zur Erstellung von Barcodes Java-Bibliotheken von Drittanbietern. Im Folgenden finden Sie die verwendeten Klassen und öffentlichen Methoden. Die Klassen befinden sich im Paket `AltovaBarcodeExtension.jar`, das im Ordner `<ProgramFilesFolder>\Altova\Common2025\jar` gespeichert ist.

Die verwendeten Java-Bibliotheken befinden sich in Unterordnern des Ordners *<ProgramFilesFolder>*\Altova\Common2025\jar:

- barcode4j\barcode4j.jar (Website: http://barcode4j.sourceforge.net/)
- zxing\core.jar (Website: http://code.google.com/p/zxing/)

Die Lizenzdateien befinden sich ebenfalls in den entsprechenden Ordnern.

Java Virtual Machine

Um die Barcode-Funktionen verwenden zu können, muss auf Ihrem Rechner eine Java Virtual Machine zur Verfügung stehen und mit der Bit-Version der Altova-Applikation übereinstimmen (32-Bit oder 64-Bit). Der Pfad zu dieser Machine wird, wie unten angeführt, gefunden.

- Wenn Sie ein Altova Desktop-Produkt verwenden, versucht die Altova-Applikation, den Pfad zur Java Virtual Machine automatisch zu ermitteln. Dazu wird zuerst (i) die Windows Registry und dann (ii) die **JAVA_HOME**-Umgebungsvariable gelesen. Sie können im Dialogfeld "Optionen" der Applikation auch einen benutzerdefinierten Pfad hinzufügen. Dieser Eintrag hat Vorrang vor allen anderen automatisch ermittelten Java VM-Pfaden.
- Wenn Sie ein Altova Server-Produkt auf einem Windows-Rechner ausführen, wird der Pfad zur Java Virtual Machine zuerst aus der Windows Registry ausgelesen. Falls dies nicht gelingt, wird die **JAVA_HOME**-Umgebungsvariable verwendet.
- Wenn Sie ein Altova Server-Produkt auf einem Linux- oder macOS-Rechner ausführen, sollten Sie sicherstellen, dass die **JAVA_HOME**-Umgebungsvariable ordnungsgemäß definiert ist und dass sich die Java Virtual Machine-Bibliothek (unter Windows die Datei **jvm.dll**) entweder im Verzeichnis **\bin\server** oder **\bin\client** befindet.

XSLT-Beispiel zum Generieren von Barcode

Im Folgenden finden Sie ein XSLT-Beispiel, in dem gezeigt wird, wie Barcode-Funktionen in einem XSLT-Stylesheet verwendet werden.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:fn="http://www.w3.org/2005/xpath-functions"
   xmlns:altova="http://www.altova.com"
   xmlns:altovaext="”http://www.altova.com/xslt-extensions”
   xmlns:altovaext-barcode="java:com.altova.extensions.barcode.BarcodeWrapper"
   xmlns:altovaext-barcode-
property="java:com.altova.extensions.barcode.BarcodePropertyWrapper">
   <xsl:output method="html" encoding="UTF-8" indent="yes"/>
   <xsl:template match="/">
      <html>
         <head><title/></head>
         <body>
            <img alt="barcode" src="data:image/png;base64"/>
         </body>
      </html>
      <xsl:result-document
         href="barcode.png"
         method="text" encoding="base64tobinary" >
         <xsl:variable name="barcodeObject"
            select="altovaext-barcode:newInstance('Code39',string('some value'),
```

```
            96,0, (altovaext-barcode-property:new( 'setModuleWidth', 25.4 div 96 *
2 ) ) )"/>
         <xsl:value-of select="xs:base64Binary(xs:hexBinary(string(altovaext-
barcode:generateBarcodePngAsHexString($barcodeObject)) ))"/>
      </xsl:result-document>
   </xsl:template>
</xsl:stylesheet>
```

XQuery-Beispiel zum Generieren eines QR-Codes

Im Folgenden finden Sie ein XQuery-Beispiel, in dem gezeigt wird, wie Sie mit Hilfe von Barcode-Funktionen ein QR-Code-Bild generieren können.

```
declare variable $lines := unparsed-text-
lines('https://info.healthministry.gv.at/data/timeline-cases-provinces.csv', 'utf-8');
declare variable $main := map:merge(tokenize(head($lines), ';')!map{.:position()});
declare variable $data := map:merge(tail($lines)!array{tokenize(., ';')}!map{?($main?Name):
[?($main?Date), xs:integer(?($main?ConfirmedCasesProvinces)) - xs:integer(?($main?
Recovered))]}, map{'duplicates':'combine'});
declare variable $chart_img := altovaext:chart(map{'width': 1900, 'height': 600}, map:for-
each($data, function($k, $v){[$k, $v?1!substring-before(., 'T'), $v?2][$k != 'Austria']}));
<img src="{$chart_img}"/>
(:$main,  $data,:)
```

Das Paket `com.altova.extensions.barcode`

Das Paket `com.altova.extensions.barcode` wird zum Generieren der meisten der Barcode-Typen verwendet.

Die folgenden Klassen werden verwendet:

```
public class BarcodeWrapper
   static BarcodeWrapper newInstance( String name, String msg, int dpi, int orientation,
BarcodePropertyWrapper[] arrProperties )
   double getHeightPlusQuiet()
   double getWidthPlusQuiet()
   org.w3c.dom.Document generateBarcodeSVG()
   byte[] generateBarcodePNG()
   String generateBarcodePngAsHexString()
```

`public class` **`BarcodePropertyWrapper`** *Dient zum Speichern der Barcode-Eigenschaften, die zu einem späteren Zeitpunkt dynamisch definiert werden*

```
    BarcodePropertyWrapper( String methodName, String propertyValue )
   BarcodePropertyWrapper( String methodName, Integer propertyValue )
   BarcodePropertyWrapper( String methodName, Double propertyValue )
   BarcodePropertyWrapper( String methodName, Boolean propertyValue )
   BarcodePropertyWrapper( String methodName, Character propertyValue )
   String getMethodName()
   Object getPropertyValue()
```

`public class AltovaBarcodeClassResolver` `Registriert die Klasse com.altova.extensions.barcode.proxy.zxing.QRCodeBean zusätzlich zu den vom org.krysalis.barcode4j.DefaultBarcodeClassResolver registrierten Klassen` *für die* `qrcode` *Bean.*

Das Paket `com.altova.extensions.barcode.proxy.zxing`

Das Paket `com.altova.extensions.barcode.proxy.zxing` wird zum Generieren des QRCode Barcodetyps verwendet.

Die folgenden Klassen werden verwendet:

`Klasse` **`QRCodeBean`**

- *Erweitert* `org.krysalis.barcode4j.impl.AbstractBarcodeBean`
- *Erstellt ein* `AbstractBarcodeBean` Interface für `com.google.zxing.qrcode.encoder`

```
void generateBarcode(CanvasProvider canvasImp, String msg)
void setQRErrorCorrectionLevel(QRCodeErrorCorrectionLevel level)
BarcodeDimension calcDimensions(String msg)
double getVerticalQuietZone()
double getBarWidth()
```

`Klasse` **`QRCodeErrorCorrectionLevel`** *Fehlerkorrekturebene für den* `QRCode`

```
static QRCodeErrorCorrectionLevel byName(String name)
"L" = ~7% correction
"M" = ~15% correction
"H" = ~25% correction
"Q" = ~30% correction
```

10.2.2 Diverse Erweiterungsfunktionen

Es gibt in Programmiersprachen wie Java und C# eine Reihe von fertigen Funktionen, die nicht als XQuery / XPath 2.0- oder XSLT-Funktionen zur Verfügung stehen. Ein gutes Beispiel dafür sind die mathematischen in Java verfügbaren Funktionen wie z.B. `sin()` und `cos()`. Stünden diese Funktionen für die Erstellung von XSLT Stylesheets und XQuery-Abfragen zur Verfügung, würde sich der Einsatzbereich von Stylesheets und Abfragen erweitern und die Erstellung von Stylesheets wäre viel einfacher. Der in einer Reihe von Altova-Produkten verwendete XSLT- und XQuery-Prozessor von Altova unterstützt die Verwendung von Erweiterungsfunktionen in Java[530] und .NET[539] sowie MSXSL Skripts für XSLT[546]. MSXSL-Skripts für XSLT[546] und die Altova-Erweiterungsfunktionen[434]. In diesem Abschnitt wird beschrieben, wie Sie Erweiterungsfunktionen und MSXSL-Skripts in Ihren XSLT Stylesheets und XQuery-Dokumenten verwenden können. Diese Beschreibungen finden Sie in den folgenden Abschnitten:

- Java-Erweiterungsfunktionen[530]
- .NET-Erweiterungsfunktionen[539]
- MSXSL-Skripts für XSLT[546]

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

Voraussetzungen

Damit die Erweiterungsfunktionen unterstützt werden, muss auf dem Rechner, auf dem die XSLT-Transformation oder die XQuery-Ausführung stattfindet, eine Java Runtime-Umgebung (zum Aufrufen der Java-Funktionen) und ein .NET Framework 2.0 (Mindestvoraussetzung für Zugriff auf .NET-Funktionen) installiert sein oder es muss Zugriff auf eine solche bestehen.

10.2.2.1 Java-Erweiterungsfunktionen

Eine Java-Erweiterungsfunktion kann in einem XPath- oder XQuery-Ausdruck verwendet werden, um einen Java-Konstruktor oder eine Java-Methode (statisch oder Instanz) aufzurufen.

Ein Feld in einer Java-Klasse wird als Methode ohne Argument betrachtet. Bei einem Feld kann es sich um ein statisches Feld oder eine Instanz handeln. Wie man Felder aufruft, wird in den entsprechenden Unterabschnitten zu statischen Feldern und Instanzen beschrieben.

Dieser Abschnitt enthält die folgenden Unterabschnitte:

Beachten Sie die folgenden Punkte

- Wenn Sie ein Altova Desktop-Produkt verwenden, versucht die Altova-Applikation, den Pfad zur Java Virtual Machine automatisch zu ermitteln. Dazu wird zuerst (i) die Windows Registry und dann (ii) die **JAVA_HOME**-Umgebungsvariable gelesen. Sie können im Dialogfeld "Optionen" der Applikation auch einen benutzerdefinierten Pfad hinzufügen. Dieser Eintrag hat Vorrang vor allen anderen automatisch ermittelten Java VM-Pfaden.
- Wenn Sie ein Altova Server-Produkt auf einem Windows-Rechner ausführen, wird der Pfad zur Java Virtual Machine zuerst aus der Windows Registry ausgelesen. Falls dies nicht gelingt, wird die **JAVA_HOME**-Umgebungsvariable verwendet.
- Wenn Sie ein Altova Server-Produkt auf einem Linux- oder macOS-Rechner ausführen, sollten Sie sicherstellen, dass die **JAVA_HOME**-Umgebungsvariable ordnungsgemäß definiert ist und dass sich die Java Virtual Machine-Bibliothek (unter Windows die Datei **jvm.dll**) entweder im Verzeichnis **\bin\server** oder **\bin\client** befindet.

Form der Erweiterungsfunktion

Die Erweiterungsfunktion im XPath/XQuery-Ausdruck muss die folgenden Form haben `präfix:fname()`.

- Der Teil `präfix:` kennzeichnet die Erweiterungsfunktion als Java-Funktion, indem er die Erweiterungsfunktion mit einer in-scope Namespace-Deklaration verknüpft, deren URI mit `java:` beginnen muss (*Beispiele siehe unten*). Die Namespace-Deklaration sollte eine Java-Klasse bezeichnen, z.B:
  `xmlns:myns="java:java.lang.Math"`. Sie könnte aber auch einfach lauten:
  `xmlns:myns="java"` (ohne Doppelpunkt), wobei die Identifizierung der Java-Klasse dem `fname()` Teil der Erweiterungsfunktion überlassen bleibt.

- Der Teil `fname()` identifiziert die aufgerufene Java-Methode und liefert die Argumente für die Methode (*Beispiele siehe unten*). Wenn die durch das `prefix:` Teil identifizierte Namespace URI jedoch keine Java-Klasse bezeichnet (*siehe vorheriger Punkt),* dann sollte die Java-Klasse im `fname()` Teil vor der Klasse identifiziert werden und von der Klasse durch einen Punkt getrennt sein (*siehe zweites XSLT-Beispiel unten*).

Anmerkung: Die aufgerufene Klasse muss sich unter dem Klassenpfad des Rechners befinden.

XSLT-Beispiel

Hier sehen Sie zwei Beispiele dafür, wie eine statische Methode aufgerufen werden kann. Im ersten Beispiel ist der Klassenname (`java.lang.Math`) in der Namespace URI enthalten und darf daher nicht im `fname()` Teil enthalten sein. Im zweiten Beispiel liefert der `prefix:` Teil das Präfix `java:`, während der `fname()` Teil die Klasse sowie die Methode identifiziert.

```
<xsl:value-of xmlns:jMath="java:java.lang.Math"
              select="jMath:cos(3.14)" />

<xsl:value-of xmlns:jmath="java"
              select="jmath:java.lang.Math.cos(3.14)" />
```

Die in der Erweiterungsfunktion (im Beispiel oben `cos()`) angegebene Methode muss mit dem Namen einer öffentlichen statischen Methode in der angegebenen Java-Klasse (im Beispiel oben `java.lang.Math`) übereinstimmen.

XQuery-Beispiel

Hier sehen Sie ein XQuery-Beispiel, das dem XSLT-Beispiel oben ähnlich ist:

```
<cosine xmlns:jMath="java:java.lang.Math">
   {jMath:cos(3.14)}
</cosine>
```

Benutzerdefinierte Java-Klassen

Wenn Sie Ihre eigenen Java-Klassen erstellt haben, werden die Methoden in diesen Klassen unterschiedlich aufgerufen, je nachdem: (i) ob die Klassen über eine JAR-Datei oder eine Klassendatei aufgerufen werden, und (ii) ob sich diese Dateien (JAR oder Klasse) im aktuellen Verzeichnis befinden (im selben Verzeichnis wie das XSLT- oder XQuery-Dokument) oder nicht. Wie Sie diese Dateien finden, wird in den Abschnitten Benutzerdefinierte Klassendateien 531 und Benutzerdefinierte Jar-Dateien 535 beschrieben. Pfade zu Klassendateien, die sich nicht im aktuellen Verzeichnis befinden, und Pfade zu allen JAR-Dateien müssen jedoch angegeben werden.

10.2.2.1.1 Benutzerdefinierte Klassendateien

Wenn der Zugriff über eine Klassendatei erfolgt, gibt es vier Möglichkeiten:

- Die Klassendatei befindet sich in einem Paket. Die XSLT-oder XQuery-Datei befindet sich im selben Ordner wie das Java-Paket. (*Siehe Beispiel unten* 532.)
- Die Klassendatei befindet sich nicht in einem Paket. Die XSLT-oder XQuery-Datei befindet sich im selben Ordner wie die Klassendatei. (*Siehe Beispiel unten* 533.)
- Die Klassendatei befindet sich in einem Paket. Die XSLT-oder XQuery-Datei befindet sich in irgendeinem beliebig gewählten Ordner. (*Siehe Beispiel unten* 533.)
- Die Klassendatei befindet sich nicht in einem Paket. Die XSLT-oder XQuery-Datei befindet sich in irgendeinem beliebig gewählten Ordner. (*Siehe Beispiel unten* 534.)

Gesetzt der Fall, die Klassendatei befindet sich nicht in einem Paket, sondern im selben Ordner wie das XSLT- oder XQuery-Dokument, so muss der Dateipfad nicht angegeben werden, da alle Klassen im Ordner gefunden werden. Die Syntax zum Identifizieren einer Klasse lautet:

```
java:classname
```

wobei

`java`: angibt, dass eine benutzerdefinierte Java-Funktion aufgerufen wird; (Java-Klassen im aktuellen Verzeichnis werden standardmäßig geladen)
`classname` der Name der Klasse der erforderlichen Methode ist

die Klasse in einer Namespace URI identifiziert wird und der Namespace einem Methodenaufruf als Präfix vorangestellt wird.

Klassendatei in einem Paket, XSLT/XQuery-Datei befindet sich im selben Ordner wie das Java-Paket

Im Beispiel unten wird die Methode `getVehicleType()` der Klasse `Car` des Pakets `com.altova.extfunc` aufgerufen. Das Paket `com.altova.extfunc` befindet sich im Ordner `JavaProject`. Die XSLT-Datei befindet sich ebenfalls im Ordner `JavaProject`.

```
<xsl:stylesheet version="2.0"
       xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
       xmlns:xs="http://www.w3.org/2001/XMLSchema"
       xmlns:fn="http://www.w3.org/2005/xpath-functions"
       xmlns:car="java:com.altova.extfunc.Car" >
<xsl:output exclude-result-prefixes="fn car xsl fo xs"/>

<xsl:template match="/">
   <a>
   <xsl:value-of select="car:getVehicleType()"/>
   </a>
</xsl:template>

</xsl:stylesheet>
```


Die Klassendatei wird referenziert, die XSLT/XQuery-Datei befindet sich im selben Ordner wie die Klassendatei

Im Beispiel unten wird die Methode `getVehicleType()` der Klasse `Car` des Pakets `com.altova.extfunc` aufgerufen. Angenommen, (i) die Klassendatei `Car` class befindet sich im folgenden Ordner: `JavaProject/com/altova/extfunc` und (ii) dieser Ordner ist der aktuelle Ordner im Beispiel unten. Die XSLT-Datei befindet sich ebenfalls im Ordner `JavaProject/com/altova/extfunc`.

```
<xsl:stylesheet version="2.0"
       xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
       xmlns:xs="http://www.w3.org/2001/XMLSchema"
       xmlns:fn="http://www.w3.org/2005/xpath-functions"
       xmlns:car="java:Car" >
<xsl:output exclude-result-prefixes="fn car xsl fo xs"/>

<xsl:template match="/">
   <a>
   <xsl:value-of select="car:getVehicleType()"/>
   </a>
</xsl:template>

</xsl:stylesheet>
```

Die Klassendatei befindet sich in einem Paket, die XSLT/XQuery-Datei befindet sich in einem beliebigen Ordner

Im Beispiel unten wird die Methode `getCarColor()` der Klasse `Car` des Pakets `com.altova.extfunc` aufgerufen. Das Paket `com.altova.extfunc` befindet sich im Ordner `JavaProject`. Die XSLT-Datei befindet sich in einem beliebigen Ordner. In diesem Fall muss der Pfad des Pakets mit der URI als Abfragestring definiert werden. Die Syntax lautet:

```
java:classname[?path=uri-of-classfile]
```

wobei

`java`: angibt, dass eine benutzerdefinierte Java-Funktion aufgerufen wird
`uri-of-classfile` die URI der Klassendatei ist
`classname` der Name der Klasse der benötigten Methode ist

die Klasse in einer Namespace URI identifiziert wird und der Namespace einem Methodenaufruf als Präfix vorangestellt wird. Im Beispiel unten sehen Sie, wie eine Klassendatei aufgerufen wird, die sich in einem anderen als dem aktuellen Verzeichnis befindet.

```
<xsl:stylesheet version="2.0"
       xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
       xmlns:xs="http://www.w3.org/2001/XMLSchema"
       xmlns:fn="http://www.w3.org/2005/xpath-functions"
       xmlns:car="java:com.altova.extfunc.Car?path=file:///C:/JavaProject/" >

<xsl:output exclude-result-prefixes="fn car xsl xs"/>
```

```
<xsl:template match="/">
   <xsl:variable name="myCar" select="car:new('red')" />
   <a><xsl:value-of select="car:getCarColor($myCar)"/></a>
</xsl:template>

</xsl:stylesheet>
```

Die Klassendatei wird referenziert, die XSLT/XQuery-Datei befindet sich in einem beliebigen Ordner

Im Beispiel unten wird die Methode `getCarColor()` der Klasse `Car` aufgerufen. Angenommen, die Klassendatei `Car` befindet sich im Ordner `C:/JavaProject/com/altova/extfunc`. Die XSLT-Datei befindet sich in einem beliebigen Ordner. Der Pfad der Klassendatei muss dann in der Namespace-URI als Abfragestring definiert werden. Die Syntax lautet:

```
java:classname[?path=<uri-of-classfile>]
```

wobei

`java:` angibt, dass eine benutzerdefinierte Java-Funktion aufgerufen wird
`uri-of-classfile` die URI der Klassendatei ist
`classname` der Name der Klasse der benötigten Methode ist

die Klasse in einer Namespace URI identifiziert wird und der Namespace einem Methodenaufruf als Präfix vorangestellt wird. Im Beispiel unten sehen Sie, wie eine Klassendatei aufgerufen wird, die sich in einem anderen als dem aktuellen Verzeichnis befindet.

```
<xsl:stylesheet version="2.0"
       xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
       xmlns:xs="http://www.w3.org/2001/XMLSchema"
       xmlns:fn="http://www.w3.org/2005/xpath-functions"
       xmlns:car="java:Car?path=file:///C:/JavaProject/com/altova/extfunc/" >

<xsl:output exclude-result-prefixes="fn car xsl xs"/>

<xsl:template match="/">
   <xsl:variable name="myCar" select="car:new('red')" />
   <a><xsl:value-of select="car:getCarColor($myCar)"/></a>
</xsl:template>

</xsl:stylesheet>
```

Anmerkung: Wenn ein Pfad über eine Erweiterungsfunktion angegeben wird, wird er zum ClassLoader hinzugefügt.

10.2.2.1.2 Benutzerdefinierte Jar-Dateien

JAR-Dateien
Wenn der Zugriff über eine JAR-Datei erfolgt, muss die URI der JAR-Datei mit Hilfe der folgenden Syntax definiert werden:

```
xmlns:classNS="java:classname?path=jar:uri-of-jarfile!/"
```

Die Methode wird anschließend durch Verwendung des Präfix der Namespace URI aufgerufen, der die Klasse bezeichnet: `classNS:method()`

wobei im obigen Beispiel:

`java:` angibt, dass eine Java-Funktion aufgerufen wird
`classname` der Name der Klasse der benutzerdefinierten Klasse ist
`?` das Trennzeichen zwischen dem Klassennamen und dem Pfad ist
`path=jar:` angibt, dass es sich um einen Pfad zu einer JAR-Datei handelt
`uri-of-jarfile` die URI der jar-Datei angibt
`!/` das Trennzeichen am Ende des Pfades ist
`classNS:method()` der Aufruf der Methode ist

Alternativ dazu kann der Klassenname mit dem Methodenaufruf angegeben werden. Hier sehen Sie zwei Beispiele für die Syntax:

```
xmlns:ns1="java:docx.layout.pages?path=jar:file:///c:/projects/docs/docx.jar!/"
ns1:main()

xmlns:ns2="java?path=jar:file:///c:/projects/docs/docx.jar!/"
ns2:docx.layout.pages.main()
```

Hier sehen Sie ein komplettes XSLT-Beispiel, in dem eine JAR-Datei verwendet wird, um eine Java-Erweiterungsfunktion aufzurufen.:

```
<xsl:stylesheet version="2.0"
       xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
       xmlns:xs="http://www.w3.org/2001/XMLSchema"
       xmlns:fn="http://www.w3.org/2005/xpath-functions"
       xmlns:car="java?path=jar:file:///C:/test/Car1.jar!/" >
<xsl:output exclude-result-prefixes="fn car xsl xs"/>

<xsl:template match="/">
       <xsl:variable name="myCar" select="car:Car1.new('red')" />
       <a><xsl:value-of select="car:Car1.getCarColor($myCar)"/></a>
</xsl:template>

<xsl:template match="car"/>

</xsl:stylesheet>
```

Anmerkung: Wenn ein Pfad über eine Erweiterungsfunktion angegeben wird, wird er zum ClassLoader hinzugefügt.

10.2.2.1.3 Java: Konstruktoren

Eine Erweiterungsfunktion kann dazu verwendet werden, um einen Java-Konstruktor aufzurufen. Alle Konstruktoren werden mit der Pseudofunktion `new()` aufgerufen.

Wenn das Ergebnis eines Java-Konstruktors implizit in XPath/XQuery-Datentypen konvertiert werden kann[539], dann gibt die Java-Erweiterungsfunktion eine Sequenz zurück, bei der es sich um einem XPath/XQuery-Datentyp handelt. Wenn das Ergebnis eines Konstruktoraufrufs nicht in einen passenden XPath/XQuery-Datentyp konvertiert werden kann, dann erstellt der Konstruktor ein wrapped Java-Objekt mit einem Typ, der den Namen der Klasse hat, die dieses Java-Objekt zurückgibt. Wenn z.B. ein Konstruktor für die Klasse `java.util.Date` aufgerufen wird (`java.util.Date.new()`), so wird ein Objekt vom Typ `java.util.Date` zurückgegeben. Das lexikalische Format des zurückgegebenen Objekts stimmt unter Umständen nicht mit dem lexikalischen Format des XPath-Datentyps überein und der Wert müsste daher in das lexikalische Format des erforderlichen XPath-Datentyps und anschließend in den erforderlichen XPath-Datentyp konvertiert werden.

Ein von einem Konstruktor erstelltes Java-Objekt kann für zwei Zwecke verwendet werden:

- Es kann einer Variable zugewiesen werden:
```
<xsl:variable name="currentdate" select="date:new()"
xmlns:date="java:java.util.Date" />
```
- Es kann an eine Erweiterungsfunktion übergeben werden (*siehe Instanzmethode und Instanzfelder*[537]):
```
<xsl:value-of select="date:toString(date:new())" xmlns:date="java:java.util.Date" />
```

10.2.2.1.4 Java: Statische Methoden und statische Felder

Eine statische Methode wird direkt über ihren Java-Namen und durch Angabe der Argumente für die Methode aufgerufen. Statische Felder (Methoden, die keine Argumente haben), wie z.B. die Konstantenwertfelder `E` und `PI` werden ohne Angabe eines Arguments aufgerufen.

XSLT-Beispiele

Hier sehen Sie einige Beispiele dafür, wie statische Methoden und Felder aufgerufen werden können:

```
<xsl:value-of xmlns:jMath="java:java.lang.Math"
              select="jMath:cos(3.14)" />

<xsl:value-of xmlns:jMath="java:java.lang.Math"
              select="jMath:cos( jMath:PI() )" />

<xsl:value-of xmlns:jMath="java:java.lang.Math"
              select="jMath:E() * jMath:cos(3.14)" />
```

Beachten Sie, dass die Erweiterungsfunktionen die Form `prefix:fname()` haben. Das Präfix ist in allen drei Fällen `jMath:`. Es ist mit der Namespace URI `java:java.lang.Math` verknüpft. (Die Namespace URI muss mit `java:`. beginnen. In den obigen Beispielen wurde es um den Klassennamen erweitert (`java.lang.Math`).) Der Teil `fname()` der Erweiterungsfunktionen muss mit dem Namen der öffentlichen Klasse (z.B.

`java.lang.Math`) gefolgt vom Namen einer öffentlichen statischen Methode mit ihrem/ihren Argument(en) (wie z.B. `(3.14)`) oder einem öffentlichen statischen Feld (z.B. `PI()`) übereinstimmen.

In den obigen Beispielen wurde der Klassenname in die Namespace URI inkludiert. Wäre sie nicht in der Namespace URI enthalten, müsste sie in den `fname()` Teil der Erweiterungsfunktion inkludiert werden. Z.B:

```
<xsl:value-of xmlns:java="java:"
              select="java:java.lang.Math.cos(3.14)" />
```

XQuery-Beispiel

Ein ähnliches Beispiel in XQuery wäre:

```
<cosine xmlns:jMath="java:java.lang.Math">
   {jMath:cos(3.14)}
</cosine>
```

10.2.2.1.5 Java: Instanzmethoden und Instanzfelder

Bei einer Instanzmethode wird als erstes Argument eines Methodenaufrufs ein Java-Objekt an die Methode übergeben. Ein solches Java-Objekt würde normalerweise mit Hilfe einer Erweiterungsfunktion (z.B. eines Konstruktoraufrufs) oder eines Stylesheet-Parameters/einer Stylesheet-Variablen erstellt. Ein XSLT-Beispiel dafür wäre:

```
<xsl:stylesheet version="1.0" exclude-result-prefixes="date"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:date="java:java.util.Date"
  xmlns:jlang="java:java.lang">
  <xsl:param name="CurrentDate" select="date:new()"/>
  <xsl:template match="/">
      <enrollment institution-id="Altova School"
                date="{date:toString($CurrentDate)}"
                type="{jlang:Object.toString(jlang:Object.getClass( date:new() ))}">
      </enrollment>
  </xsl:template>
</xsl:stylesheet>
```

Im Beispiel oben wird der Wert des Node `enrollment/@type` folgendermaßen erstellt:

1. Es wird ein Objekt mit einem Konstruktor für die Klasse `java.util.Date` (mit dem Konstruktor `date:new()`) erstellt.
2. Dieses Java-Objekt wird als das Argument der Methode `jlang.Object.getClass` übergeben.
3. Das mit der Methode `getClass` abgerufene Objekt wird als das Argument an die Methode `jlang.Object.toString` übergeben.

Das Ergebnis (der Wert von `@type`) ist ein String, der den Wert `java.util.Date` hat.

Ein Instanzfeld unterscheidet sich theoretisch insofern von einer Instanzmethode, als es sich nicht um ein Java-Objekt per se handelt, das als Argument an das Instanzfeld übergeben wird. Stattdessen wird ein Parameter oder eine Variable als Argument übergeben. Der Parameter/die Variable kann allerdings selbst den Wert

enthalten, der von einem Java-Objekt zurückgegeben wird. So erhält z.B. der Parameter `CurrentDate` den Wert, der von einem Konstruktor für die Klasse `java.util.Date` zurückgegeben wird. Dieser Wert wird anschließend als Argument an die Instanzmethode `date:toString` übergeben, um den Wert von `/enrollment/@date` bereitzustellen.

10.2.2.1.6 Datentypen: XPath/XQuery in Java

Wenn von einem XPath/XQuery-Ausdruck aus eine Java-Funktion aufgerufen wird, spielt der Datentyp der Argumente der Funktion eine wichtige Rolle, welche von mehreren Java-Klassen desselben Namens aufgerufen wird.

In Java gelten die folgenden Regeln:

- Wenn es mehr als eine Java-Methode mit demselben Namen gibt, jede aber eine andere Anzahl von Argumenten als die andere(n) hat, so wird die Java-Methode ausgewählt, die der Anzahl der Argumente im Funktionsaufruf am ehesten entspricht.
- Die XPath/XQuery-Datentypen "string", "number" und "boolean" (*siehe Liste unten*) werden implizit in einen entsprechenden Java-Datentyp konvertiert. Wenn der bereitgestellte XPath/XQuery-Datentyp in mehr als einen Java-Typ konvertiert werden kann (z.B: `xs:integer`), so wird jener Java-Typ ausgewählt, der für die ausgewählte Methode deklariert wurde. Wenn die aufgerufene Java-Methode z.B. `fx(decimal)` und der bereitgestellte XPath/XQuery-Datentyp `xs:integer` ist, so wird `xs:integer` in den Java-Datentyp `decimal` konvertiert.

In der Tabelle unten sehen Sie eine Liste der impliziten Konvertierungen der XPath/XQuery-Datentypen "string", "number" und "boolean"in Java-Datentypen.

`xs:string`	`java.lang.String`
`xs:boolean`	`boolean` (primitive), `java.lang.Boolean`
`xs:integer`	int, long, short, byte, float, double und die Wrapper-Klassen davon wie z.B. `java.lang.Integer`
`xs:float`	`float` (primitive), `java.lang.Float`, `double` (primitive)
`xs:double`	`double` (primitive), `java.lang.Double`
`xs:decimal`	`float` (primitive), `java.lang.Float`, `double`(primitive), `java.lang.Double`

Die oben aufgelisteten Subtypen von XML-Schema-Datentypen (die in XPath und XQuery verwendet werden) werden ebenfalls in den/die Java-Typ(en), der/die dem übergeordneten Subtyp entsprechen, konvertiert.

In einigen Fällen ist es nicht möglich, auf Basis der verfügbaren Informationen die richtige Java-Methode auszuwählen. Nehmen Sie als Beispiel den folgenden Fall.

- Das bereitgestellte Argument ist ein `xs:untypedAtomic` Wert 10 und ist für die Methode `mymethod(float)` bestimmt.

- Es gibt jedoch eine weitere Methode in der Klasse, die ein Argument eines anderen Datentyps erhält: `mymethod(double)`.
- Da die Methodennamen dieselben sind und der bereitgestellte Typ (`xs:untypedAtomic`) sowohl in `float` als auch `double` korrekt konvertiert werden könnte, kann es geschehen, dass `xs:untypedAtomic` in `double` anstelle von `float` konvertiert wird.
- Infolgedessen handelt es sich dann bei der ausgewählten Methode nicht um die benötigte Methode, sodass nicht das erwartete Ergebnis erzielt wird. Als Umgehungslösung können Sie eine benutzerdefinierte Methode mit einem anderen Namen erstellen und diese Methode verwenden.

Typen, die in der Liste oben nicht enthalten sind (z.B. `xs:date`), werden nicht konvertiert und generieren einen Fehler. Beachten Sie jedoch, dass es in einigen Fällen unter Umständen möglich ist, den benötigten Java-Typ mittels eines Java-Konstruktors zu erstellen.

10.2.2.1.7 Datentypen: Java in XPath/XQuery

Wenn eine Java-Methode einen Wert zurückgibt und der Datentyp des Werts "string", "numeric" oder "boolean" ist, wird anschließend in den entsprechenden XPath/XQuery-Typ konvertiert. So werden z.B. die Java-Datentypen `javaa.lang.Boolean` und `boolean` in `xsd:boolean` konvertiert.

Von Funktionen zurückgegebene eindimensionale Arrays werden zu einer Sequenz erweitert. Mehrdimensionale Arrays werden nicht konvertiert und sollten daher in einen Wrapper gesetzt werden.

Wenn ein wrapped Java-Objekt oder ein Datentyp zurückgegeben wird, bei dem es sich nicht um den Typ "string", "numeric" oder "boolean" handelt, können Sie sicherstellen, dass die Konvertierung in den benötigten XPath/XQuery-Typ erfolgt, indem Sie zuerst eine Java-Methode (e.g `toString`) verwenden, um das Java-Objekt in einen String zu konvertieren. In XPath/XQuery kann der String geändert werden, damit er der lexikalischen Darstellung des benötigten Typs entspricht, und anschließend z.B. mit Hilfe des Ausdrucks `cast as` in den benötigten Typ konvertiert werden.

10.2.2.2 .NET-Erweiterungsfunktionen

Wenn Sie auf einem Windows-Rechner mit der .NET-Plattform arbeiten, können Sie Erweiterungsfunktionen verwenden, die in jeder beliebigen der .NET-Sprachen geschrieben wurden (z.B. C#). Eine .NET Erweiterungsfunktion kann in einem XPath- oder XQuery-Ausdruck verwendet werden, um einen Konstruktor, eine Eigenschaft oder Methode (statische oder Instanz) in einer .NET-Klasse aufzurufen.

Eine Eigenschaft einer .NET-Klasse wird mit der Syntax `get_PropertyName()` aufgerufen.

Dieser Abschnitt ist in die folgenden Unterabschnitte gegliedert:

Form der Erweiterungsfunktion

Die Erweiterungsfunktion im XPath/XQuery-Ausdruck muss die folgende Form haben `präfix:fname()`.

- Der Teil `präfix:` ist mit einer URI verknüpft, die die benötigte .NET-Klasse definiert.
- Der Teil `fname()` identifiziert den Konstruktor, die Eigenschaft oder die Methode (statisch oder Instanz) innerhalb der .NET-Klasse und liefert alle gegebenenfalls benötigten Argumente.

- Die URI muss mit `clitype:` beginnen (welches die Funktion als .NET-Erweiterungsfunktion kennzeichnet).
- Die Form `prefix:fname()` der Erweiterungsfunktion kann mit Systemklassen und mit Klassen in einer geladenen Assembly verwendet werden. Wenn eine Klasse allerdings geladen werden muss, müssen zusätzliche Parameter mit den benötigten Informationen bereitgestellt werden.

Parameter

Zum Laden einer Assembly werden die folgenden Parameter verwendet:

`asm`	Der Name der zu ladenden Assembly
`ver`	Die Versionsnummer: eine Maximalzahl von vier Ganzzahlen, die durch Punkte getrennt sind
`sn`	Das Key Token des Strong Name der Assembly (16 Hex-Stellen).
`from`	Eine URI gibt den Pfad der zu ladenden Assembly (DLL) an. Wenn die URI relativ ist, ist sie relativ zum XSLT- oder XQuery-Dokument. Wenn dieser Parameter vorhanden ist, werden alle anderen Parameter ignoriert.
`partialname`	Der partielle Name der Assembly. Er wird für `Assembly.LoadWith.PartialName()` bereitgestellt, welches versuchen wird, die Assembly zu laden. Wenn `partialname` vorhanden ist, werden alle anderen Parameter ignoriert.
`loc`	Die Locale, z.B. `en-US`. Die Standardeinstellung ist `neutral`

Wenn die Assembly aus einer DLL geladen werden soll, verwenden Sie den `from` Parameter und lassen Sie den `sn` Parameter weg. Wenn die Assembly aus dem Global Assembly Cache (GAC) geladen werden soll, verwenden Sie den `sn` Parameter und lassen Sie den `from` Parameter weg.

Vor dem ersten Parameter muss ein Fragezeichen eingefügt werden. Parameter müssen durch ein Semikolon getrennt werden. Der Wert des Parameternamens wird durch ein Ist-Gleich-Zeichen angegeben (*siehe Beispiele unten*).

Beispiele für Namespace-Deklarationen

Ein Beispiel für eine Namespace Deklaration in XSLT, die die Systemklasse `System.Environment`: identifiziert.

```
xmlns:myns="clitype:System.Environment"
```

Ein Beispiel für eine Namespace Deklaration in XSLT, die die zu ladende Klasse als `Trade.Forward.Scrip`: identifiziert.


```
xmlns:myns="clitype:Trade.Forward.Scrip?asm=forward;version=10.6.2.1"
```

Ein Beispiel für eine Namespace-Deklaration in XQuery, die die Systemklasse `MyManagedDLL.testClass` identifiziert. Es werden zwei Klassen unterschieden:

1. Wenn die Assembly aus dem GAC geladen wird:
   ```
   declare namespace cs="clitype:MyManagedDLL.testClass?asm=MyManagedDLL;
   ver=1.2.3.4;loc=neutral;sn=b9f091b72dccfba8";
   ```
2. Wenn die Assembly aus der DLL geladen wird (vollständige und partielle Referenzen unten):
   ```
   declare namespace cs="clitype:MyManagedDLL.testClass?from=file:///C:/Altova
   Projects/extFunctions/MyManagedDLL.dll;

declare namespace cs="clitype:MyManagedDLL.testClass?from=MyManagedDLL.dll;
```

XSLT-Beispiel

Hier sehen Sie ein vollständiges XSLT-Beispiel, in dem Funktionen in der Systemklasse `System.Math`: aufgerufen werden:

```
<xsl:stylesheet version="2.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:fn="http://www.w3.org/2005/xpath-functions">
   <xsl:output method="xml" omit-xml-declaration="yes" />
   <xsl:template match="/">
      <math xmlns:math="clitype:System.Math">
         <sqrt><xsl:value-of select="math:Sqrt(9)"/></sqrt>
         <pi><xsl:value-of select="math:PI()"/></pi>
         <e><xsl:value-of select="math:E()"/></e>
         <pow><xsl:value-of select="math:Pow(math:PI(), math:E())"/></pow>
      </math>
   </xsl:template>
</xsl:stylesheet>
```

Die Namespace-Deklaration für das Element `math` verknüpft das Präfix `math:` mit der URI `clitype:System.Math`. Der Beginn der URI `clitype:` gibt an, dass danach entweder eine Systemklasse oder eine geladene Klasse definiert wird. Das Präfix `math:` im XPath-Ausdruck verknüpft die Erweiterungsfunktionen mit der URI (und durch Erweiterung der Klasse) `System.Math`. Die Erweiterungsfunktionen identifizieren Methoden in der Klasse `System.Math` und stellen Argumente bereit, wo dies erforderlich ist.

XQuery-Beispiel

Hier sehen Sie ein XQuery-Beispielfragment ähnlich dem XSLT-Beispiel oben:

```
<math xmlns:math="clitype:System.Math">
   {math:Sqrt(9)}
</math>
```

Wie beim XSLT-Beispiel weiter oben identifiziert die Namespace-Deklaration die .NET-Klasse, in diesem Fall eine Systemklasse. Der XQuery-Ausdruck identifiziert die aufzurufenden Methode und liefert das Argument.

10.2.2.2.1 .NET: Konstruktoren

Eine Erweiterungsfunktion kann verwendet werden, um einen .NET-Konstruktor aufzurufen. Alle Konstruktoren werden mit der Pseudofunktion `new()` aufgerufen. Wenn es mehrere Konstruktoren für eine Klasse gibt, wird der Konstruktor ausgewählt, der der Anzahl der bereitgestellten Argumente am ehesten entspricht. Wenn kein passender Konstruktor gefunden wird, der den bereitgestellten Argumenten entspricht, wird die Fehlermeldung `'No constructor found'` zurückgegeben.

Konstruktoren, die XPath/XQuery-Datentypen zurückgeben

Wenn das Ergebnis eines .NET-Konstruktors implizit in XPath/XQuery-Datentypen konvertiert werden kann539, gibt die .NET-Erweiterungsfunktion eine Sequenz zurück, bei der es sich um einen XPath/XQuery-Datentyp handelt.

Konstruktoren, die .NET-Objekte zurückgeben

Wenn das Ergebnis eines .NET-Konstruktoraufrufs nicht in einen passenden XPath/XQuery-Datentyp konvertiert werden kann, erstellt der Konstruktor ein wrapped .NET-Objekt mit einem Typ, der der Name der Klasse ist, die dieses Objekt zurückgibt. Wenn z.B. ein Konstruktor für die Klasse `System.DateTime` aufgerufen wird (mit `System.DateTime.new()`), so wird ein Objekt mit dem Typ `System.DateTime` zurückgegeben.

Das lexikalische Format des zurückgegebenen Objekts stimmt unter Umständen nicht mit dem lexikalischen Format eines erforderlichen XPath-Datentyps überein. In solchen Fällen müsste der zurückgegebene Wert: (i) in das lexikalische Format des benötigten XPath-Datentyps konvertiert werden; und (ii) in den erforderlichen XPath-Datentyp konvertiert werden.

Ein von einem Konstruktor erstelltes .NET-Objekt kann für drei Zwecke verwendet werden:

- Es kann innerhalb einer Variable verwendet werden:
  ```
  <xsl:variable name="currentdate" select="date:new(2008, 4, 29)"
  xmlns:date="clitype:System.DateTime" />
  ```
- Es kann an eine Erweiterungsfunktion übergeben werden (*siehe Instanzmethode und Instanzfelder*537):
  ```
  <xsl:value-of select="date:ToString(date:new(2008, 4, 29))"
  xmlns:date="clitype:System.DateTime" />
  ```
- Es kann in einen String, eine Zahl oder einen Booleschen Ausdruck konvertiert werden:
- ```
  <xsl:value-of select="xs:integer(date:get_Month(date:new(2008, 4, 29)))"
  xmlns:date="clitype:System.DateTime" />
  ```

10.2.2.2.2 .NET: Statische Methoden und statische Felder

Eine statische Methode wird direkt über ihren Namen und durch Angabe der Argumente für die Methode aufgerufen. Der im Aufruf verwendete Name muss exakt mit einer öffentlichen statischen Methode in der angegebenen Klasse übereinstimmen. Wenn der Methodenname und die Anzahl der in der Funktion angegebenen Argumente mit mehr als einer Methode in einer Klasse übereinstimmen, werden die Typen der


  ```

bereitgestellten Argumente nach der besten Übereinstimmung überprüft. Wenn keine eindeutig passende Methode gefunden werden kann, wird ein Fehler ausgegeben.

**Anmerkung:** Ein Feld in einer .NET-Klasse wird als Methode ohne Argument betrachtet. Eine Eigenschaft wird mit der Syntax `get_PropertyName()` aufgerufen.

### Beispiele

Ein XSLT-Beispiel, in dem Sie einen Methodenaufruf mit einem Argument (`System.Math.Sin(arg)`) sehen:

```
<xsl:value-of select="math:Sin(30)" xmlns:math="clitype:System.Math"/>
```

Ein XSLT-Beispiel, in dem Sie einen Aufruf eines Felds (wird als Methode ohne Argument betrachtet) sehen (`System.Double.MaxValue()`):

```
<xsl:value-of select="double:MaxValue()" xmlns:double="clitype:System.Double"/>
```

Ein XSLT-Beispiel, in dem Sie einen Aufruf einer Eigenschaft (Syntax ist `get_PropertyName()`) (`System.String()`) sehen:

```
<xsl:value-of select="string:get_Length('my string')"
xmlns:string="clitype:System.String"/>
```

Ein XQuery-Beispiel, in dem Sie einen Aufruf einer Methode mit einem Argument (`System.Math.Sin(arg)`) sehen:

```
<sin xmlns:math="clitype:System.Math">
   { math:Sin(30) }
</sin>
```

10.2.2.2.3 .NET: Instanzmethoden und Instanzfelder

Bei einer Instanzmethode wird als erstes Argument des Methodenaufrufs ein .NET-Objekt an die Methode übergeben. Dieses .NET-Objekt wird normalerweise mit Hilfe einer Erweiterungsfunktion (z.B. durch einen Konstruktoraufruf) oder einen Stylesheet-Parameter/eine Stylesheet-Variable erstellt. Ein XSLT-Beispiel dieser Art wäre:

```
<xsl:stylesheet version="2.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:fn="http://www.w3.org/2005/xpath-functions">
   <xsl:output method="xml" omit-xml-declaration="yes"/>
   <xsl:template match="/">
      <xsl:variable name="releasedate"
         select="date:new(2008, 4, 29)"
         xmlns:date="clitype:System.DateTime"/>
      <doc>
```

```
        <date>
           <xsl:value-of select="date:ToString(date:new(2008, 4, 29))"
              xmlns:date="clitype:System.DateTime"/>
        </date>
        <date>
           <xsl:value-of select="date:ToString($releasedate)"
              xmlns:date="clitype:System.DateTime"/>
        </date>
     </doc>
  </xsl:template>
</xsl:stylesheet>
```

Im Beispiel oben wird ein `System.DateTime` Konstruktor (`new(2008, 4, 29)`) verwendet, um ein .NET-Objekt vom Typ `System.DateTime` zu erstellen. Diese Objekt wird zweimal erstellt, einmal als Wert der Variablen `releasedate`, ein zweites Mal als das erste und einzige Argument der Methode `System.DateTime.ToString()`. Die Instanzmethode `System.DateTime.ToString()` wird zwei Mal aufgerufen, beide Male mit dem `System.DateTime` Konstruktor (`new(2008, 4, 29)`) als erstem und einzigem Argument. In einer dieser Instanzen wird die Variable `releasedate` verwendet, um das .NET-Objekt abzurufen.

Instanzmethoden und Instanzfelder

Der Unterschied zwischen einer Instanzmethode und einem Instanzfeld ist ein theoretischer. In einer Instanzmethode wird ein .NET-Objekt direkt als Argument übergeben; in einem Instanzfeld wird stattdessen ein Parameter oder eine Variable übergeben - auch wenn der Parameter bzw. die Variable selbst ein .NET-Objekt enthalten kann. So enthält z.B. die Variable `releasedate` im Beispiel oben ein .NET-Objekt und es ist diese Variable, die als das Argument von `ToString()` an den zweiten `date` Elementkonstruktor übergeben wird. Die `ToString()` Instanz im ersten `date` Element ist daher eine Instanzmethode, während die zweite als Instanzfeld betrachtet wird. Das in beiden Instanzen erzeugte Ergebnis ist jedoch dasselbe.

10.2.2.2.4 Datentypen: XPath/XQuery in .NET

Wenn in einem XPath/XQuery-Ausdruck eine .NET-Erweiterungsfunktion verwendet wird, spielen die Datentypen der Argumente der Funktion eine wichtige Rolle bei der Entscheidung, welche der vielen .NET-Methoden mit demselben Namen aufgerufen werden soll.

In .NET gelten die folgenden Regeln:

- Wenn es mehr als eine Methode mit demselben Namen in einer Klasse gibt, so stehen nur die Methoden zur Auswahl, die dieselbe Anzahl von Argumenten wie der Funktionsaufruf haben.
- Die XPath/XQuery-Datentypen "string", "number" und "boolean" (*siehe Liste unten*) werden implizit in einen entsprechenden .NET-Datentyp konvertiert. Wenn der bereitgestellte XPath/XQuery-Datentyp in mehr als einen .NET-Typ konvertiert werden kann (z.B: `xs:integer`), so wird jener .NET-Typ ausgewählt, der für die ausgewählte Methode deklariert wurde. Wenn die aufgerufene .NET-Methode z.B. `fx(double)` und der bereitgestellte XPath/XQuery-Datentyp `xs:integer` ist, so wird `xs:integer` in den .NET-Datentyp `double`

In der Tabelle unten sehen Sie eine Liste der impliziten Konvertierungen der XPath/XQuery-Datentypen "string", "number" und "boolean"in .NET-Datentypen.

xs:string	StringValue, string
xs:boolean	BooleanValue, bool
xs:integer	IntegerValue, decimal, long, integer, short, byte, double, float
xs:float	FloatValue, float, double
xs:double	DoubleValue, double
xs:decimal	DecimalValue, decimal, double, float

Die oben aufgelisteten Subtypen von XML-Schema-Datentypen (die in XPath und XQuery verwendet werden) werden ebenfalls in den/die .NET-Typ(en), der/die dem übergeordneten Subtyp entsprechen, konvertiert.

In einigen Fällen ist es nicht möglich, auf Basis der verfügbaren Informationen die richtige .NET-Methode auszuwählen. Nehmen Sie als Beispiel den folgenden Fall.

- Das bereitgestellte Argument ist ein `xs:untypedAtomic` Wert 10 und ist für die Methode `mymethod(float)` bestimmt.
- Es gibt jedoch eine weitere Methode in der Klasse, die ein Argument eines anderen Datentyps erhält: `mymethod(double)`.
- Da die Methodennamen dieselben sind und der bereitgestellte Typ (`xs:untypedAtomic`) sowohl in `float` als auch `double` korrekt konvertiert werden könnte, kann es geschehen, dass `xs:untypedAtomic` in `double` anstelle von `float` konvertiert wird.
- Infolgedessen handelt es sich dann bei der ausgewählten Methode nicht um die benötigte Methode, sodass nicht das erwartete Ergebnis erzielt wird. Als Umgehungslösung können Sie eine benutzerdefinierte Methode mit einem anderen Namen erstellen und diese Methode verwenden.

Typen, die in der Liste oben nicht enthalten sind (z.B. `xs:date`), werden nicht konvertiert und generieren einen Fehler.

10.2.2.2.5 Datentypen: .NET in XPath/XQuery

Wenn eine .NET-Methode einen Wert zurückgibt und der Datentyp des Werts "string", "numeric" oder "boolean" ist, wird er anschließend in den entsprechenden XPath/XQuery-Typ konvertiert. So wird z.B. der .NET-Datentyp `decimal` in `xsd:decimal` konvertiert.

Wenn ein .NET-Objekt oder ein Datentyp zurückgegeben wird, bei dem es sich nicht um den Typ "string", "numeric" oder "boolean" handelt, können Sie sicherstellen, dass die Konvertierung in den benötigten XPath/XQuery-Typ erfolgt, indem Sie zuerst eine .NET-Methode (z.B. `System.DateTime.ToString()`) verwenden, um das .NET-Objekt in einen String zu konvertieren. In XPath/XQuery kann der String geändert werden, damit er der lexikalischen Darstellung des benötigten Typs entspricht, und anschließend z.B. mit Hilfe des Ausdrucks `cast as` in den benötigten Typ konvertiert werden.

10.2.2.3 MSXSL-Skripts für XSLT

Das Element `<msxsl:script>` enthält benutzerdefinierte Funktionen und Variablen, die von XPath-Ausdrücken im XSLT-Stylesheet aufgerufen werden können. Das Element `<msxsl:script>` ist ein Element der obersten Ebene, d.h. es muss ein Child-Element von `<xsl:stylesheet>` oder `<xsl:transform>` sein.

Das Element `<msxsl:script>` muss sich im Namespace `urn:schemas-microsoft-com:xslt` (*siehe Beispiel unten*) befinden.

Scripting-Sprache und Namespace

Die im Block verwendete Scripting-Sprache wird im Attribut `language` des Elements `<msxsl:script>` definiert und der für Funktionsaufrufe von XPath-Ausdrücken aus zu verwendende Namespace wird durch das Attribut `implements-prefix` (*siehe unten*) identifiziert.

```
<msxsl:script language="scripting-language implements-prefix="user-namespace-prefix">

   function-1 or variable-1
   ...
   function-n or variable-n

</msxsl:script>
```

Das Element `<msxsl:script>` interagiert mit der Windows Scripting Runtime. Daher können nur Sprachen, die auf Ihrem Rechner installiert sind, im Element `<msxsl:script>` verwendet werden. **Um MXSL Scripts verwenden zu können muss die Plattform .NET Framework 2.0 oder höher installiert sein.** Folglich können die .NET Scripting Sprachen innerhalb des Elements `<msxsl:script>` verwendet werden.

Das Attribut `language` akzeptiert dieselben Werte wie das Attribut `language` des HTML `<script>` Elements. Wenn das Attribut `language` nicht definiert ist, wird als Standardsprache Microsoft JScript verwendet.

Das Attribut `implements-prefix` erhält einen Wert, der ein Präfix eines deklarierten in-scope Namespace ist. Bei diesem Namespace handelt es sich normalerweise um einen Benutzer-Namespace, der für eine Funktionsbibliothek reserviert ist. Alle Funktionen und Variablen, die im Element `<msxsl:script>` definiert sind, werden sich im Namespace befinden, der durch das im Attribut `implements-prefix` definierte Präfixe identifiziert wird. Wenn eine Funktion von einem XPath-Ausdruck aus aufgerufen wird, muss sich der vollständig qualifizierte Funktionsname im selben Namespace wie die Funktionsdefinition befinden.

Beispiel

Hier sehen Sie ein Beispiel für ein vollständiges XSLT Stylesheet, in dem eine Funktion verwendet wird, die in einem `<msxsl:script>` Element definiert ist.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:fn="http://www.w3.org/2005/xpath-functions"
   xmlns:msxsl="urn:schemas-microsoft-com:xslt"
   xmlns:user="http://mycompany.com/mynamespace">
```



```
  <msxsl:script language="VBScript" implements-prefix="user">
    <![CDATA[
    ' Input: A currency value: the wholesale price
    ' Returns: The retail price: the input value plus 20% margin,
    ' rounded to the nearest cent
    dim a as integer  = 13
    Function AddMargin(WholesalePrice) as integer
      AddMargin = WholesalePrice * 1.2 + a
    End Function
  ]]>
  </msxsl:script>

  <xsl:template match="/">
    <html>
      <body>
        <p>
          <b>Total Retail Price =
            $<xsl:value-of select="user:AddMargin(50)"/>
          </b>
          <br/>
          <b>Total Wholesale Price =
            $<xsl:value-of select="50"/>
          </b>
        </p>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

Datentypen

Die Werte von Parametern, die an und aus dem Script-Block heraus übergeben werden, sind auf XPath-Datentypen beschränkt. Diese Einschränkung gilt nicht für Daten, die zwischen Funktionen und Variablen innerhalb des Script-Blocks übergeben werden.

Assemblies

Eine Assembly kann über das Element **`msxsl:assembly`** in das Script importiert werden. Die Assembly wird über einen Namen oder eine URL identifiziert. Die Assembly wird beim Kompilieren des Stylesheet importiert. Hier sehen Sie ein einfaches Beispiel, wie das Element `msxsl:assembly` zu verwenden ist.

```
<msxsl:script>
   <msxsl:assembly name="myAssembly.assemblyName" />
   <msxsl:assembly href="pathToAssembly" />

   ...

</msxsl:script>
```

Der Assembly-Name kann ein vollständiger Name sein, wie z.B.:

```
"system.Math, Version=3.1.4500.1 Culture=neutral PublicKeyToken=a46b3f648229c514"
```

oder ein Kurzname wie z.B. `"myAssembly.Draw"`.

Namespaces

Namespaces können mit dem Element **`msxsl:using`** deklariert werden. Auf diese Art können Assembly-Klassen ohne ihre Namespaces in das Script geschrieben werden, wodurch Sie sich das mühsame Eintippen ersparen. Hier sehen Sie, wie das Element `msxsl:using` verwendet wird, um Namespaces zu deklarieren.

```
<msxsl:script>
   <msxsl:using namespace="myAssemblyNS.NamespaceName" />

   ...

</msxsl:script>
```

Der Wert des `namespace` Attributs ist der Name des Namespace.

Index

I

J

K

L

M

N

P

R

S

U

V

W

X

Z