Altova RaptorXML Server 2024

Manual del usuario y referencia

Altova RaptorXML Server 2024
Manual del usuario y referencia

Todos los derechos reservados. Ningún fragmento de esta publicación podrá ser reproducido de manera alguna (ya sea de forma gráfica, electrónica o mecánica, fotocopiado, grabado o reproducido en sistemas de almacenamiento y recuperación de información) sin el consentimiento expreso por escrito de su autor/editor.

Los productos a los que se hace referencia en este documento pueden ser marcas registradas de sus respectivos propietarios. El autor y editor no afirman ser propietarios de dichas marcas registradas.

Durante la elaboración de este documento se tomaron todas las precauciones necesarias para prevenir errores. Sin embargo, el autor y editor no se responsabilizan de los errores u omisiones que pudiese contener el documento ni de los posibles daños o perjuicios derivados del uso del contenido de este documento o de los programas y código fuente que vengan con el documento. Bajo ninguna circunstancia se podrá considerar al autor y editor responsables de la pérdida de beneficios ni de cualquier otro daño y perjuicio derivado directa o indirectamente del uso de este documento.

Fecha de publicación: 2023

© 2017-2023 Altova GmbH

Contenido

1 Introducción 9

2 Acerca de RaptorXML Server 10

2.1 Ediciones e interfaces 11
2.2 Requisitos del sistema 15
2.3 Características 17
2.4 Especificaciones compatibles 19
2.5 Cambios notables 21

3 Configurar RaptorXML Server 22

3.1 Instalación y configuración en Windows 23
3.1.1 Instalación en Windows 23
3.1.2 Instalación en Windows Server Core 24
3.1.3 Instalar LicenseServer (Windows) 27
3.1.4 Configuración de red y de servicios (Windows) 28
3.1.5 Asignar una licencia a RaptorXML Server (Windows) 29
3.2 Instalación y configuración en Linux 34
3.2.1 Instalación en Linux 34
3.2.2 Instalar LicenseServer (Linux) 36
3.2.3 Asignar licencias a RaptorXML Server (Linux) 37
3.3 Instalación y configuración en macOS 40
3.3.1 Instalación en macOS 40
3.3.2 Instalar LicenseServer (macOS) 41
3.3.3 Asignar licencias a RaptorXML Server (macOS) 42
3.4 Actualizar RaptorXML Server 45
3.5 Migrar RaptorXML Server a un equipo nuevo 46

4 Procedimientos generales 47

4.1 Catálogos XML ... 48
4.1.1 Funcionamiento de los catálogos ... 48
4.1.2 Estructura de los catálogos en RaptorXML Server ... 50
4.1.3 Personalizar catálogos ... 51
4.1.4 Variables para ubicaciones de sistemas Windows ... 53
4.2 Recursos globales ... 55
4.3 Problemas de seguridad ... 57

5 Interfaz de la línea de comandos (ILC) 58

5.1 Comandos para validar XML, DTD, XSD ... 59
5.1.1 valxml-withdtd (xml) ... 59
5.1.2 valxml-withxsd (xsi) ... 63
5.1.3 valdtd (dtd) ... 71
5.1.4 valxsd (xsd) ... 75
5.2 Comandos para comprobar el formato ... 83
5.2.1 wfxml ... 83
5.2.2 wfdtd ... 87
5.2.3 wfany ... 91
5.3 Comandos XQuery ... 96
5.3.1 xquery ... 96
5.3.2 xqueryupdate ... 104
5.3.3 valxquery ... 114
5.3.4 valxqueryupdate ... 121
5.4 Comandos XSLT ... 128
5.4.1 xslt ... 128
5.4.2 valxslt ... 137
5.5 Comandos JSON/Avro ... 144
5.5.1 avroextractschema ... 144
5.5.2 json2xml ... 148
5.5.3 jsonschema2xsd ... 153
5.5.4 valavro (avro) ... 158

5.5.5 valavrojson (avrojson) ... 162
5.5.6 valavroschema (avroschema) ... 166
5.5.7 valjsonschema (jsonschema) ... 170
5.5.8 valjson (json) ... 174
5.5.9 wfjson ... 179
5.5.10 xml2json ... 184
5.5.11 xsd2jsonschema ... 189
5.6 Comandos XML Signature ... 196
5.6.1 xmlsignature-sign ... 196
5.6.2 xmlsignature-verify ... 201
5.6.3 xmlsignature-update ... 204
5.6.4 xmlsignature-remove ... 207
5.7 Comandos generales ... 209
5.7.1 valany ... 209
5.7.2 script ... 210
5.7.3 help ... 211
5.8 Comandos de localización ... 213
5.8.1 exportresourcestrings ... 213
5.8.2 setdeflang ... 215
5.9 Comandos de licencias ... 217
5.9.1 licenseserver ... 217
5.9.2 assignlicense (solo Windows) ... 219
5.9.3 verifylicense (solo Windows) ... 220
5.10 Comandos de gestión ... 222
5.10.1 install ... 223
5.10.2 uninstall ... 223
5.10.3 start ... 224
5.10.4 setdeflang ... 225
5.10.5 licenseserver ... 226
5.10.6 assignlicense (Windows only) ... 228
5.10.7 verifylicense (Windows only) ... 229
5.10.8 createconfig ... 230
5.10.9 exportresourcestrings ... 231
5.10.10 debug ... 233

5.10.11 help ... 234
5.10.12 version ... 235
5.11 Opciones ... 236
5.11.1 Catálogos, recursos globales, archivos ZIP ... 236
5.11.2 Mensajes, errores, ayuda, tiempo de espera y versión ... 237
5.11.3 Procesamiento ... 238
5.11.4 XML ... 239
5.11.5 XSD ... 240
5.11.6 XQuery ... 243
5.11.7 XSLT ... 245
5.11.8 JSON/Avro ... 247
5.11.9 Firmas XML ... 248

6 APIs de servidor: HTTP REST, COM/.NET, Java 252

6.1 Interfaz cliente HTTP REST ... 254
6.1.1 Preparar el servidor ... 255
6.1.2 Solicitudes cliente ... 267
6.1.3 Ejemplo en C# para API REST ... 287
6.2 APIs COM y .NET ... 291
6.2.1 Interfaz COM ... 291
6.2.2 Ejemplo de COM: VBScript ... 291
6.2.3 Interfaz NET ... 293
6.2.4 Ejemplo de .NET: C# ... 295
6.2.5 Ejemplo de .NET: Visual Basic .NET ... 297
6.3 API de Java ... 300
6.3.1 Resumen de la interfaz Java ... 300
6.3.2 Ejemplo de proyecto Java ... 301
6.4 Referencia sobre las APIs de servidor ... 303
6.4.1 Interfaces/Clases ... 303
6.4.2 Enumeraciones ... 355

7 APIs de motor: Python y .NET 367

7.1 Licencias ... 369

7.2 API de Python ... 371
7.2.1 Versiones de la API de Python ... 372
7.2.2 RaptorXML Server como paquete Python ... 374
7.2.3 Depurar scripts de Python del lado servidor ... 377
7.2.4 Depurar scripts de Python en Visual Studio Code ... 378
7.2.5 Preguntas frecuentes ... 379
7.3 API de .NET Framework ... 381

8 Gestor de esquemas 382

8.1 Ejecutar el gestor de esquemas ... 386
8.2 Categorías de estado ... 389
8.3 Aplicar parches o instalar un esquema ... 391
8.4 Desinstalar o restaurar esquemas ... 393
8.5 Interfaz de la línea de comandos (ILC) ... 394
8.5.1 help ... 394
8.5.2 info ... 395
8.5.3 initialize ... 395
8.5.4 install ... 396
8.5.5 list ... 396
8.5.6 reset ... 397
8.5.7 uninstall ... 398
8.5.8 update ... 399
8.5.9 upgrade ... 399

9 Información adicional 401

9.1 Códigos de salida ... 402
9.2 Sugerencias sobre ubicación de esquemas ... 403

10 Información sobre motores 404

10.1 Información sobre motores XSLT y XQuery ... 405
10.1.1 XSLT 1.0 ... 405
10.1.2 XSLT 2.0 ... 405
10.1.3 XSLT 3.0 ... 407

10.1.4 XQuery 1.0 408
10.1.5 XQuery 3.1 412
10.2 Funciones XSTL y XPath/XQuery 414
10.2.1 Funciones de extensión de Altova 415
10.2.2 Funciones de extensión varias 511

Índice **530**

1 Introducción

Altova RaptorXML Server (en adelante RaptorXML) es el rapidísimo motor XML y XBRL de tercera generación de Altova, optimizado para los estándares más recientes y para entornos de informática en paralelo. RaptorXML es compatible con múltiples plataformas y aprovecha la omnipresencia actual de equipos multinúcleo para ofrecer rapidísimas funciones de procesamiento de datos XML y XBRL.

Nota: las funciones de procesamiento XBRL solamente están disponibles en RaptorXML+XBRL Server (no están disponibles en RaptorXML Server).

Esta documentación

La presente documentación está incluida en la aplicación y también está disponible en el sitio web de Altova. La presente documentación se divide en varias secciones:

- Acerca de RaptorXML Server 10
- Instalar RaptorXML Server 47
- Interfaz de la línea de comandos (ILC) 58
- APIs de servidor: HTTP REST, COM/.NET, Java 252
- APIs de motor: Python y .NET 367
- Información adicional 401
- Información sobre motores 404

Sitio web de Altova: Servidor de validación XML, Validación XML

Última actualización: 17.10.2023

2 Acerca de RaptorXML Server

Ediciones y sistemas operativos

Altova ofrece dos ediciones diferentes de RaptorXML, diseñadas para satisfacer diferentes requisitos. Estas dos ediciones se describen en el apartado Ediciones e interfaces 11. RaptorXML está disponible para Windows, Linux y macOS. Para más información consulte el apartado Requisitos del sistema 15.

Características y especificaciones compatibles

RaptorXML ofrece funciones de validación XML, transformación XSLT y ejecución de XQuery dotadas de numerosas y potentes opciones. Para ver la lista de características y funciones clave de RaptorXML, consulte el apartado Características 17. En el apartado Especificaciones compatibles 19 se enumeran todas las especificaciones con las que cumple RaptorXML. Para más información visite el sitio web de Altova.

© 2017-2023 Altova GmbH

2.1 Ediciones e interfaces

Ediciones

Altova ofrece dos ediciones distintas de RaptorXML:

- ***RaptorXML Server:*** un rapidísimo motor de procesamiento XML compatible con XML, XML Schema, XSLT, XPath y XQuery, entre otros estándares.
- ***RaptorXML+XBRL Server:*** ofrece todas las características de RaptorXML Server y funciones de procesamiento y validación compatibles con todos los estándares XBRL.

Consulte aquí[19] la lista de especificaciones compatibles[19].

Interfaces

Puede acceder a RaptorXML Server a través de varias interfaces:

- *Interfaz de la línea de comandos (ILC):* disponible para las versiones de RaptorXML para Windows, Linux y macOS
- *Interfaz de cliente HTTP REST:* usa la interfaz HTTP *REST* de RaptorXML
- *Interfaz de servidor COM/.NET (Windows):* usa las interfaces (i) COM/.NET y (ii) HTTP *REST* de RaptorXML
- *Interfaz de servidor Java (Windows, Linux, macOS):* usa (i) la API Java y (ii) la interfaz HTTP *REST* de RaptorXML
- Interfaz de Altova XMLSpy: se puede acceder a RaptorXML desde la interfaz de usuario de Altova XMLSpy
- *Interfaz de servidor Python:* usa (i) un paquete wheel Python de RaptorXML en su entorno Python y (ii) la API de Python de RaptorXML en su script Python. Esto permite usar las funciones de RaptorXML en scripts de Python junto con paquetes de Python de terceros.
- *Interfaz .NET (Windows):* usa (i) un DLL de RaptorXML y (ii) la API de .NET de RaptorXML para crear aplicaciones .NET independientes que usen las funciones de RaptorXML.

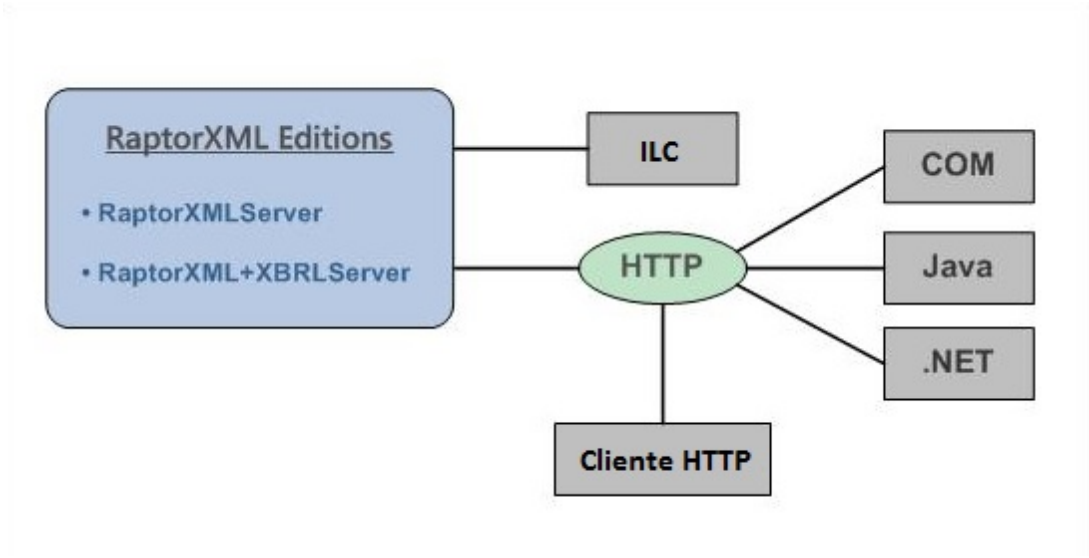
Estas siete interfaces se organizan en cuatro grupos:

- Interfaz de la línea de comandos (ILC)[58]
- APIs de servidor: HTTP, COM/.NET, Java[252]
- APIs de motor: Python y .NET[367]
- Altova XMLSpy[13]

ILC, APIs de servidor y Altova XMLSpy

El diagrama que aparece a continuación muestra cómo se accede a la ILC, las APIs de servidor y a Altova XMLSpy a través de las diferentes interfaces.

RaptorXML Server define una interfaz HTTP REST que los clientes usan para enviar trabajos de validación al servidor. Los clientes pueden acceder a la interfaz HTTP REST directamente o usar las APIs de servidor de alto nivel de COM/.NET y Java. Estas APIs permiten usar fácilmente clases COM/.NET y Java para gestionar la creación y el envío de solicitudes HTTP REST. Altova XMLSpy también puede configurar para que ejecute trabajos de validación en un RaptorXML Server remoto.



Interfaz de la línea de comandos (ILC)

- RaptorXML tiene una licencia asignada en el equipo en el que está instalado; a esta instancia se accede desde la línea de comandos
- se puede instalar en Windows, Linux y macOS
- permite usar la línea de comandos [58] para validar y procesar documentos XML, XML Schema, XML Signature, XQuery y XSLT
- la versión de 3.11.5 de Python está incluida en RaptorXML y se usa al invocar un script de Python con la opción `--script`

Interfaz cliente HTTP REST

- RaptorXML tiene una licencia asignada en el equipo en el que está instalado; a esta instancia se accede desde una interfaz cliente HTTP REST [254].
- las solicitudes cliente se realizan en formato JSON. A cada solicitud se le asigna en el servidor un directorio de trabajo en el que se guardan archivos de salida. Las respuestas del servidor al cliente incluyen toda la información relevante sobre el trabajo.
- la versión de 3.11.5 de Python está incluida en RaptorXML y se usa al invocar un script de Python con la opción `--script`.

Interfaz COM/.NET

- solo disponible en Windows
- durante la instalación, RaptorXML Server se registra automáticamente como objeto de servidor COM, lo que permite invocar a RaptorXML Server desde otras aplicaciones y lenguajes de scripting compatibles con el uso de llamadas COM.
- RaptorXML tiene una licencia asignada en el equipo en el que está instalado.
- la interfaz .NET está construida como un envoltorio en torno a la interfaz COM.
- la API de servidor COM/.NET [291] de RaptorXML cuenta con objetos que se pueden usar en lenguajes de programación COM/.NET para acceder a las funciones de RaptorXML.

 © 2017-2023 Altova GmbH

- la versión de 3.11.5 de Python está incluida en RaptorXML y se usa al invocar un script de Python con la opción `--script`.

Interfaz Java

- RaptorXML tiene una licencia asignada en el equipo en el que está instalado; a esta instancia se accede desde un programa Java.
- las funciones de RaptorXML están disponibles en la API de servidor Java 300 como clases de Java que se pueden usar en programas Java.
- la versión de 3.11.5 de Python está incluida en RaptorXML y se usa al invocar un script de Python con la opción `--script`.

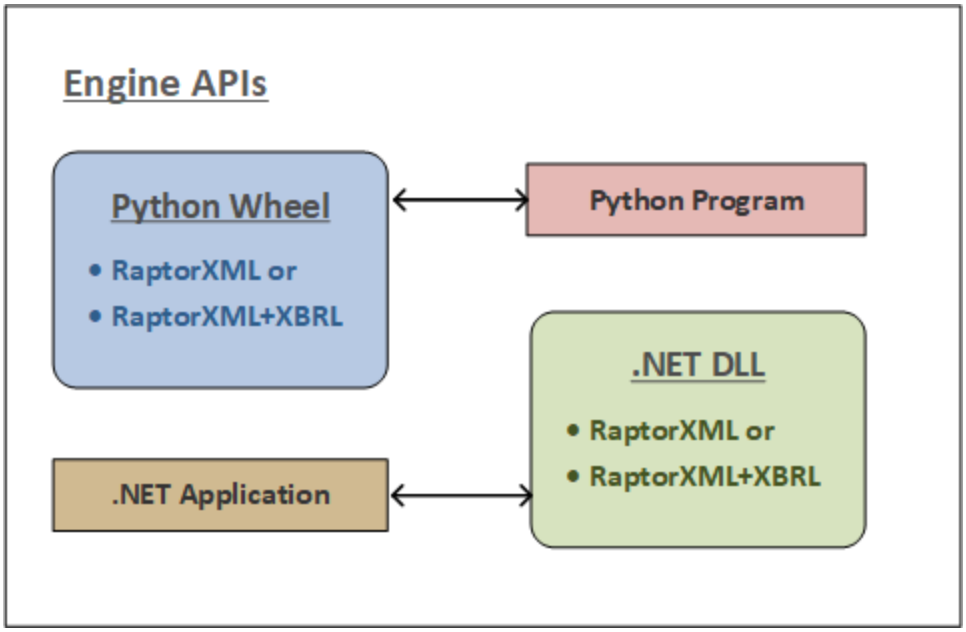
AltovaXMLSpy

- Si tiene instalado Altova XMLSpy con una licencia asignada y si XMLSpy puede acceder a RaptorXML Server a través de una red, entonces puede usar RaptorXML Server desde la interfaz gráfica de usuario de XMLSpy para validar documentos XML, además de realizar transformaciones XSLT y XQuery.
- Puede validar el documento activo o todos los documentos de una carpeta de proyecto de XMLSpy.
- Los resultados de la validación se muestran en la ventana Mensajes de la interfaz gráfica de usuario de XMLSpy.
- En XMLSpy puede (i) validar documentos o (ii) realizar transformaciones XSLT/XQuery usando los motores de XMLSpy o el servidor RaptorXML.
- Una de las principales ventajas de usar Raptor es que puede individualizar la configuración de las validaciones con un amplio abanico de opciones de validación. Asimismo, puede guardar un conjunto de opciones de Raptor como "configuración" en XMLSpy y seleccionar una de esas configuraciones ya definidas para una validación en particular. Raptor también resulta útil para validar grandes colecciones de datos.

APIs de motor

Las APIs de motor 367 son distintas de las APIs de servidor en el sentido de que RaptorXML está contenido en el archivo wheel Python y en el DLL de .NET que usan, respectivamente, los programas Python y las aplicaciones .NET (*imagen siguiente*). Estos programas y aplicaciones deben usar respectivamente la API de Python 371 y la API de .NET 381 para poder acceder a las funciones de RaptorXML.

Nota: API de Python 371 y la API de .NET 381 incluyen muchas más funciones que las que ofrecen tanto la ILC como las APIs servidor, como la posibilidad de leer documentos y manipular datos.



Interfaz de Python

- RaptorXML está disponible en un paquete wheel de Python que puede instalar en su entorno de Python 3.11.5.
- después puede escribir un programa Python que use objetos de la API de Python[371] de RaptorXML. Esta API ofrece muchas más funciones que las que ofrece la ILC y se puede combinar con las funciones de bibliotecas de terceros de su entorno Python.
- cuando se llama a las funciones de RaptorXML mediante el archivo wheel Python de RaptorXML, se comprueba que la licencia del RaptorXML instalado en ese equipo es válida antes de ejecutar el comando.

Interfaz .NET

- RaptorXML está disponible en un DLL que se puede combinar con aplicaciones compatibles con el .NET Framework. Consulte el apartado API de .NET Framework[381] para más información sobre la API.
- la API .NET[381] de RaptorXML permite acceder a RaptorXML. Ofrece muchas más funciones que la ILC de RaptorXML.
- cuando se llama a las funciones de RaptorXML mediante una aplicación .NET, se comprueba si existe una licencia válida de RaptorXML en ese equipo.

© 2017-2023 Altova GmbH

2.2 Requisitos del sistema

RaptorXML Server es compatible con estos sistemas operativos:

- *Windows*

  Windows 10, Windows 11

- *Windows Server*

  Windows Server 2012 o superior

Requisitos del sistema (Linux)

- Red Hat Enterprise Linux 7 o superior
- CentOS 7, CentOS Stream 8
- Debian 9 o superior
- Ubuntu 18.04, 20.04, 22.04
- AlmaLinux 9.0
- Rocky Linux 9.0

Requisitos previos

- Realizar la instalación como usuario **root** o como usuario con privilegios **sudo**.
- Debe desinstalar la versión anterior de RaptorXML Server antes de instalar la nueva.
- Si tiene pensado usar la función de gráficos de Altova debe tener instalada al menos una fuente en su sistema para garantizar que los gráficos se visualizan correctamente. Para ver qué fuentes tiene instaladas puede usar el comando **`fc-list`** de la biblioteca Fonconfig.
- Las bibliotecas que aparecen a continuación son un requisito previo para la instalación y ejecución de la aplicación. Si los paquetes que aparecen en esta tabla no están en su equipo Linux, ejecute el comando `yum` (o `apt-get` si procede) para instalarlos.

Requisito para	CentOS, RedHat	Debian	Ubuntu
LicenseServer	libidn o libidn2, krb5-libs, fontconfig	libidn2-0, libgssapi-krb5-2, fontconfig	libidn2-0, libgssapi-krb5-2, libfontconfig
RaptorXML Server	libidn o libidn2, krb5-libs	libidn2-0, libgssapi-krb5-2	libidn2-0, libgssapi-krb5-2

- *macOS*

  macOS 11 o superior

RaptorXML es compatible con equipos de 32 y 64 bits. Estos son los núcleos basados en conjuntos de instrucciones x86 y amd64 (x86-64): Intel Core i5, i7, XEON E5.

Para usar RaptorXML Server a través de una interfaz COM el usuario debe tener privilegios para usar la interfaz COM, es decir, para registrar la aplicación y ejecutar las aplicaciones y scripts pertinentes.

© 2017-2023 Altova GmbH

2.3 Características

RaptorXML ofrece todas las funciones que aparecen a continuación. La mayoría de las funciones corresponden a la línea de comandos y a la interfaz COM. La principal diferencia es que la interfaz COM en Windows permite construir documentos a partir de cadenas de texto con el código de aplicación o de script (en lugar de hacer referencia a archivos XML, DTD, esquemas XML, XSLT o XQuery).

Validación XML

- Valida documentos XML con esquemas XML y DTD internos o externos.
- Revisa el formato de documentos XML, DTD, XML Schema, XSLT y XQuery.

Transformaciones XSLT

- Transforma XML usando documentos XSLT 1.0, 2.0 o 3.0 suministrados por el usuario.
- Los documentos XML y XSLT se pueden suministrar en forma de archivpo (por su URL) o, en el caso de la interfaz COM, en forma de cadena de texto.
- Los resultados se devuelven en forma de archivo (en la ubicación elegida por el usuario) o, en el caso de la interfaz COM, en forma de cadena de texto.
- Los parámetros XSLT se pueden suministrar a través de la línea de comandos o de la interfaz de COM.
- Las funciones de extensión de Altova, así como las funciones de extensión Java y .NET, permiten un procesamiento más especializado. Por ejemplo, permiten crear ciertas características como gráficos y códigos de barras en los documentos de salida.

Ejecución de XQuery

- Ejecuta documentos XQuery 1.0 y 3.0.
- Los documentos XQuery y XML se pueden suministrar en forma de archivo (por su URL) o, en el caso de la interfaz COM, en forma de cadena de texto.
- Los resultados se devuelven en forma de archivo (en la ubicación elegida por el usuario) o, en el caso de la interfaz COM, en forma de cadena de texto.
- Las variables XQuery externas se pueden suministrar a través de la línea de comandos o de la interfaz de COM.
- Opciones de serialización: codificación de salida, método de codificación (es decir, si el resultado es en XML, XHTML, HTML o texto), omisión de la declaración XML y sangría.

Validación y conversión de datos JSON y Avro

- Validación de documentos de esquema JSON y Avro.
- Validación de instancias JSON con esquemas JSON y esquemas Avro.
- Validación de binarios Avro.
- Conversión de binarios Avro en esquemas Avro y de datos Avro en formato JSON.
- Conversión de datos Avro JSON en binarios Avro.

Características de alto rendimiento

- Optimizaciones de código de altísimo rendimiento.
  - Implementaciones nativas de conjuntos de instrucciones.
  - Versión de 32 bits o de 64 bits.
- Bajísima superficie de memoria.
  - Representación en memoria de XML Information Set extremadamente compacta.
  - Validación de instancias por transmisión por secuencias.
- Características compatibles con múltiples plataformas.
- Código altamente adaptable para informática en paralelo y equipos multi-CPU/multinúcleo.
- Carga, validación y procesamiento en paralelo.

Características para desarrolladores

- Avanzadas funciones de generación de informes de errores.
- Modo servidor Windows y modo demonio Unix (a través de opciones de la línea de comandos).
- Intérprete Python 3.x para scripting.
- Todas las funciones de RaptorXML en un paquete de Python para importar las funciones como biblioteca Python.
- API de .NET Framework para acceder al modelo de datos XML subyacente.
- API de COM en la plataforma Windows.
- API de Java en todas las plataformas.
- Funciones de extensión XPath, Java, .NET, XBRL, etc.
- Serialización de secuencias de datos.
- Servidor HTTP integrado con API de validación REST.

Para más información consulte el apartado Especificaciones compatibles 19 y visite el sitio web de Altova.

 © 2017-2023 Altova GmbH

2.4 Especificaciones compatibles

RaptorXML es compatible con todas estas especificaciones.

Recomendaciones del W3C

Sitio web: 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

Borradores y recomendaciones candidatas del W3C

Sitio web: World Wide Web Consortium (W3C)

- XSL Transformations (XSLT) Version 3.0 (subset)
- 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

Estándares OASIS

Sitio web: OASIS Standards

- XML Catalogs V 1.1 - OASIS Standard V1.1

Estándares JSON/Avro

Sitios web: JSON Schema y 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

© 2017-2023 Altova GmbH

2.5 Cambios notables

A continuación se indican los cambios de cada versión que podrían ser de interés.

v2024

A partir de esta versión, en la línea de comandos, la opción **`--network-timeout`** indica un valor en milisegundos (en lugar de segundos como ocurría en versiones anteriores). Esta opción se puede configurar para varios comandos y aparece en *Opciones comunes* en la descripción del comando. Para ver un ejemplo, consulte el comando valxml-withxsd (xsi)[63].

3 Configurar RaptorXML Server

En esta sección se explica cómo instalar su producto, asignarle una licencia y configurarlo. La sección se divide en varios apartados:

- Instalación en Windows 23
- Instalación en Linux 34
- Instalación en macOS 40
- Actualizar RaptorXML Server 45
- Migrar RaptorXML Server a un equipo nuevo 46

© 2017-2023 Altova GmbH

3.1 Instalación y configuración en Windows

En este apartado se describen la installación[23] y asignación de licencias[27] de RaptorXML Server en sistemas Windows.

Requisitos del sistema (Windows)

- Windows 10, Windows 11
- Windows Server 2012 o superior

Requisitos

- Debe hacer la instalación como usuario con privilegios de administrador.
- A partir de la versión 2021 no se puede instalar una versión de 32 bits de RaptorXML Server sobre una de 64 bits o viceversa. Debe (i) desinstalar la versión antigua antes de instalar la nueva o (ii) actualizar su versión a una más reciente pero de la misma versión de bits que la que ya tenía.

3.1.1 Instalación en Windows

RaptorXML Server se puede instalar en sistemas Windows. A continuación explicamos cómo instalar y configurar su producto. Para información detallada sobre partes específicas del proceso de instalación consulte los apartados correspondientes.

Instalar RaptorXML Server

Para instalar RaptorXML Server en sistemas Windows siga estos pasos:

- Como producto servidor independiente. Para instalar RaptorXML Server descargue y ejecute su instalador. Siga las instrucciones que aparecen en pantalla.
- Para instalar RaptorXML Server como parte del paquete de FlowForce Server descargue e instale el instalador de FlowForce Server. Siga las instrucciones que aparecen en pantalla y asegúrese de que marca la casilla para que también se instale RaptorXML Server.

Puede encontrar tanto el instalador de RaptorXML Server como el de FlowForce Server en el Centro de descargas de Altova (http://www.altova.com/es/download.html). Puede seleccionar el idioma de instalación en la parte inferior izquierda del asistente de instalación. Tenga en cuenta que el idioma que elija en la instalación también será el idioma predeterminado de RaptorXML Server. Puede cambiar el idioma más tarde desde la línea de comandos.

Después de la instalación encontrará el ejecutable de RaptorXML Server en esta carpeta:

```
<CarpetaArchivosPrograma>\Altova\RaptorXMLServer2024\bin\RaptorXML.exe
```

El instalador se encarga de realizar los registros necesarios para usar RaptorXML Server a través de una interfaz COM, como interfaz Java y en el entorno .NET. También registra el ejecutable de RaptorXML Server como objeto de servidor COM y añade el archivo `Altova.RaptorXML.dll` a la biblioteca de referencia .NET.

Instalación en Windows Server Core

Windows Server Core no tiene ninguna IGU y debe instalarse a través de la línea de comandos. Para más información sobre cómo instalar el equipo, consulte el apartado Instalación en Windows Server Core 24.

Desinstalar RaptorXML Server

Para desinstalar RaptorXML Server siga estos pasos:

1. Haga clic con el botón derecho en el botón de **Inicio** de Windows y seleccione **Opciones**.
2. Abra el Panel de control (empiece a escribir "Panel de control" y haga clic en la entrada sugerida).
3. En *Programas* haga clic en **Desinstalar un programa**.
4. En el Panel de control seleccione RaptorXML Server y haga clic en **Desinstalar**.

Licencia de evaluación

Durante el proceso de instalación podrá solicitar una licencia de evaluación para RaptorXML Server que dura 30 días. Una vez enviada la solicitud recibirá una licencia de evaluación en la dirección de correo electrónico que indicó al registrarse.

3.1.2 Instalación en Windows Server Core

Windows Server Core es una instalación mínima de Windows que no usa todas las características de la IGU. Para instalar RaptorXML Server en un equipo Windows Server Core:

1. Descargue el instalador de RaptorXML Server desde el Centro de descargas de Altova. Este archivo se llama **`RaptorXMLServer<version>.exe`**. Asegúrese de que escoge el ejecutable que coincide con la plataforma de su servidor (32 bits o 64 bits).
2. En un equipo Windows estándar (no en el equipo con Windows Server Core) ejecute el comando **`RaptorXMLServer<version>.exe /u`**. Este comando abre los archivos **`.msi`** en el mismo directorio que el ejecutable del instalador.
3. Copie el archivo **`.msi`** sin descomprimir en el equipo en el que está Windows Server Core.
4. Si está actualizando una versión antigua de RaptorXML Server cierre RaptorXML Server antes de ejecutar el paso siguiente.
5. Use el archivo **`.msi`** para la instalación; para ello ejecute el comando **`msiexec /i RaptorXMLServer.msi`**. Este comando inicia la instalación en Windows Server Core.

Nota: al actualizar a una versión más reciente puede conservar la configuración de RaptorXML Server usando las propiedades que explicamos en los apartados de esta sección: (i) Propiedades del servidor web 26, (ii) Propiedades del servicio SSL 26 y (iii) Propiedades del servicio 27.

Importante: conserve el archivo MSI

Debe tener en cuenta que:

- Mantenga los archivos **`.msi`** extraídos en un lugar seguro. Lo necesitará más tarde para desinstalar, reparar o modificar su instalación.
- Si quiere cambiar el nombre del archivo MSI debe hacerlo antes de instalar RaptorXML Server.

 © 2017-2023 Altova GmbH

- El nombre del archivo MSI se almacena en el registro, que es donde puede actualizar el nombre si este ha cambiado.

Registrar RaptorXML Server con LicenseServer.

Si instala RaptorXML Server por primera vez o si está actualizando el programa a la **versión siguiente**, necesita registrar RaptorXML Server con Altova LicenseServer en su red. Si está instalando una actualización menor de RaptorXML Server, el instalador reconocerá que ya registró el programa con LicenseServer, por lo que no será necesario volver a registrarlo. Sin embargo, si quiere cambiar el LicenseServer que usa RaptorXML Server en algún momento, debe registrar RaptorXML Server con License Server.

Para registrar RaptorXML Server con Altova LicenseServer después de la instalación ejecute el comando de instalación con la propiedad **REGISTER_WITH-LICENSE_SERVER**, como se ve a continuación, e indique el nombre o la dirección del LicenseServer correspondiente como valor de la propiedad, por ejemplo:

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

Para registrar RaptorXML Server con un Altova LicenseServer después de instalarlo ejecute este comando:

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

Comandos útiles

A continuación encontrará varios comandos que pueden serle útiles en el contexto de la instalación.

Para probar el valor de retorno de la instalación puede que quiera ejecutar un script parecido al que mostramos a continuación. El código de retorno estará en la variable de entorno `%errorlevel%` . El código de retorno `0` indica que la operación se ha realizado correctamente.

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

Si prefiere realizar una instalación silenciosa con un código de retorno y un registro del proceso de instalación use este comando:

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

Para modificar la instalación, ejecute:

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

Para reparar la instalación, ejecute:

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

Para desinstalar RaptorXML Server:

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

Para una desinstalación silenciosa de RaptorXML Server y obtener un registro detallado del resultado en un archivo de registro:

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

Para instalar RaptorXML Server en otro idioma (los idiomas disponibles y sus códigos son: Alemán=`de`; Español=`es`; Francés=`fr`):

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

Nota: en Windows Server Core no están disponibles los gráficos deRaptorXML Server.

3.1.2.1 Webserver Properties

Puede configurar el servidor web de RaptorXML Server usando las propiedades que aparecen a continuación. Para configurar una propiedad ejecute el comando de instalación y anexe la configuración de la propiedad:

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

Lista de propiedades

Propiedades del servidor web de RaptorXML Server:

RXML_WebServer_Host=<IP4 Address>
Use `127.0.0.1` si quiere acceder al servidor web solamente desde este equipo. Use `0.0.0.0` para hacer que el servidor web sea accesible de forma global.

RXML_WebServer_Port=<Port Number>
Indica el puerto que se usa para acceder al servidor web.

RXML_WebServer_Enabled=<0 o 1>
Seleccione:`1` para permitir escuchar en el puerto configurado actualmente. Seleccione `0` para dejar de escuchar en este puerto.

3.1.2.2 SSL-Webserver Properties

Puede configurar el servidor SSL web de RaptorXML Server usando las propiedades que aparecen a continuación. Para configurar una propiedad ejecute el comando de instalación y anexe la configuración de la propiedad:

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

Lista de propiedades

Para configurar el servidor web SSL de RaptorXML Server use estas propiedades:

RXML_SSLWebServer_Host=<IP4 Address>
Use `127.0.0.1` si quiere acceder al servidor web SSL (para una transmisión cifrada) solamente desde este equipo. Use `0.0.0.0` para hacer que el servidor web SSL sea accesible de forma global.

RXML_SSLWebServer_Port=<Port Number>
Indica el puerto que se usa para acceder al servidor web SSL (para una transmisión cifrada).

RXML_SSLWebServer_Enabled=<0 o 1>
Seleccione:`1` para permitir escuchar en el puerto configurado actualmente. Seleccione `0` para dejar de escuchar en este puerto.

 © 2017-2023 Altova GmbH

RXML_SSLWebServer_Certificate=<Path-to-certificate-file>
Ruta completa de acceso a un certificado SSL entre comillas dobles.

RXML_SSLWebServer_PrivateKey=<Path-to-private-key-file>
Ruta completa de acceso a un archivo de clave privada entre comillas.

3.1.2.3 Service Properties

Puede configurar el servicio de RaptorXML Server usando las propiedades que puede ver a continuación. Para configurar una propiedad ejecute el comando de instalación y anexe la configuración de la propiedad:

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

Lista de propiedades

Para configurar los servicios de RaptorXML Server use estas propiedades:

RXML_Service_DisplayName=<Serveice Display Name>
El nombre que aparece para ese servicio. Debe indicar este nombre entre comillas dobles.

RXML_Service_StartType=<Startup Type>
Indica cómo se inicia el servicio al inicializar el sistema. El valor puede ser uno de estos: **auto | auto-delayed | demand | disabled**.

RXML_Service_Username=<UserName>
Indica el usuario que se conecta al servicio. Use uno de estos: **LocalSystem | NT Authority\LocalService | NT Authority\NetworkService | <cualquier usuario con los derechos relevantes>**.

RXML_Service_Password=<Password>
La contraseña del usuario inicial del servicio en texto simple. (Consejo: use la interfaz del usuario del instalador para evitar introducir contraseñas en texto simple.) No necesita indicar ninguna contraseña si el nombre de usuario es uno de estos: **LocalSystem | NT Authority\LocalService | NT Authority\NetworkService**.

3.1.3 Instalar LicenseServer (Windows)

Para poder utilizar RaptorXML Server debe asignarle una licencia con un Altova LicenseServer de su red. En sistemas Windows puede instalar LicenseServer al mismo tiempo que RaptorXML Server o FlowForce Server. Si ya hay un LicenseServer instalado en su red no necesita instalar uno nuevo, a no ser que necesite actualizar la versión de LicenseServer. (*Véase el punto siguiente, versiones de LicenseServer.*)

Durante el proceso de instalación de RaptorXML Server or FlowForce Server, active o desactive la opción para instalar LicenseServer según corresponda. Tenga en cuenta que:

- Si no ha instalado Altova LicenseServer, no cambie las opciones seleccionadas por defecto. El asistente instalará la versión más reciente en el equipo donde se está ejecutando.
- Si no ha instalado Altova LicenseServer y quiere instalarlo en otro equipo, desactive la casilla *Instalar Altova LicenseServer* en este equipo y elija el botón de opción **Registrar más tarde**. En este caso deberá instalar LicenseServer por separado y registrar RaptorXML Server por más adelante.
- Si LicenseServer ya está instalado en el equipo pero se trata de una versión anterior a la indicada por el asistente para la instalación, no cambie las opciones seleccionadas por defecto (la opción predeterminada es actualizar a la versión más reciente). En este caso, el asistente de instalación actualizará la versión de su LicenseServer automáticamente. La información del registro y de la licencia se traspasa a la versión más reciente de LicenseServer.
- Si LicenseServer ya está instalado en el equipo o en la red y se trata de la misma versión que la indicada por el asistente para la instalación, entonces siga estos pasos:
  - Desactive la casilla *Instalar Altova LicenseServer en este equipo*.
  - En *Registrar este producto con* elija el LicenseServer con el que quiere registrar RaptorXML Server. También puede optar por la opción **Registrar más tarde**. Siempre puede elegir la opción **Registrar más tarde** si no quiere asociar el producto con LicenseServer y continuar con la instalación de RaptorXML Server.

Para más información sobre cómo registrar RaptorXML Server y asignarle una licencia con Altova LicenseServer consulte el apartado Licensing RaptorXML Server 29.

Versiones de LicenseServer

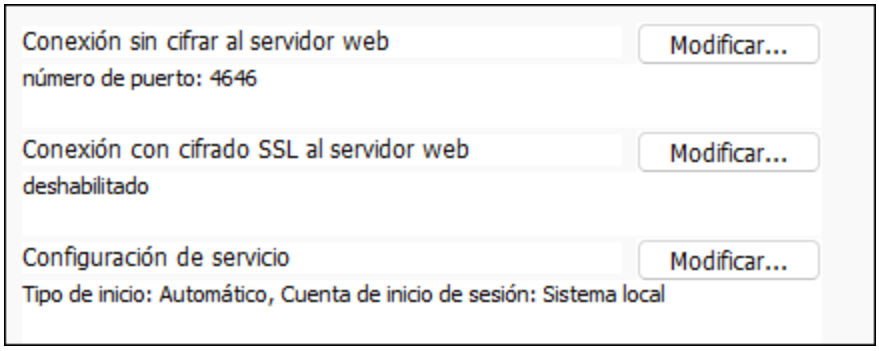
- Los productos servidor de Altova deben tener una licencia con la versión de LicenseServer correspondiente a la versión de RaptorXML Server instalada o con una versión posterior de LicenseServer.
- La versión de LicenseServer correspondiente a la versión de RaptorXML Server aparece en pantalla durante la instalación de RaptorXML Server. Puede instalar esta versión de LicenseServer junto con RaptorXML Server o puede instalar LicenseServer por separado.
- Antes de instalar una versión nueva de LicenseServer, es necesario desinstalar versiones anteriores. El programa de instalación de LicenseServer se encarga de esto automáticamente si detecta versiones más recientes en el sistema.
- Las versiones de LicenseServer son compatibles y funcionan con versiones más antiguas de RaptorXML Server.
- Si instala una versión nueva de RaptorXML Server y la versión de LicenseServer que está instalada es anterior a la que le corresponde, instale la versión más reciente que está siempre disponible en el sitio web de Altova.
- Cuando se desinstala LicenseServer, todos los datos de registro y asignación de licencias almacenados en la versión antigua de LicenseServer se guardan en una base de datos en el equipo servidor. Estos datos se importan de forma automática a la siguiente versión que se instale en el equipo.
- El número de versión de LicenseServer siempre aparece al final de la página de configuración de LicenseServer.

  Versión actual: 3.12

3.1.4 Configuración de red y de servicios (Windows)

Durante la instalación de RaptorXML Server puede configurar las opciones para acceder a RaptorXML Server a través de la red y para ejecutar RaptorXML Server como servicio de Windows (*imagen siguiente*).

 © 2017-2023 Altova GmbH



A continuación puede ver las opciones disponibles. Deje las opciones predeterminadas si le convienen o si no está seguro de cómo cambiarlas. Si quiere cambiar una opción seleccione su botón **Cambiar** (*imagen anterior*).

- El puerto que se debe usar para la comunicación sin cifrar con RaptorXML Server.
- Si se permiten conexiones seguras (con cifrado SSL) a RaptorXML Server. Si sí, en qué puerto. Por defecto las conexiones seguras están deshabilitadas. Para más información, consulte el apartado en que se explica el cifrado SSL 264.
- Configuración del servicio de Windows, que incluye:
  - El tipo de inicio puede ser automático, bajo petición, automático (inicio retrasado) o deshabilitado.
  - Qué cuenta de usuario debe usar RaptorXML Server para el servicio de Windows: *Sistema local*, *Servicio local*, *Servicio de red* u *Otro usuario*. Si selecciona *Otro usuario* podrá definir el nombre y la contraseña de este usuario (más o menos como en la consola de administración de servicios Windows). Tenga en cuenta que el usuario seleccionado debe tener derecho de lectura/escritura en `C:\ProgramData\Altova`. De lo contrario la instalación podría fallar.

Puede cambiar estas opciones una vez finalizada la instalación. Para cambiar la configuración de servicios Windows abra la consola administrativa de servicios Windows (para ello teclee `Services.msc` en la ventana de la línea de comandos) y realice ahí los cambios.

3.1.5 Asignar una licencia a RaptorXML Server (Windows)

Para poder trabajar con RaptorXML Server es necesario asignarle una licencia con Altova LicenseServer. La asignación de licencias es un proceso de dos pasos:

1. El primero consiste en **registrar RaptorXML Server** con LicenseServer desde RaptorXML Server.
2. El segundo paso consiste en **asignar una licencia** a RaptorXML Server desde LicenseServer. Descargue la versión más reciente de LicenseServer desde el sitio web de Altova e instálelo en su equipo local o en uno de los equipos de su red.

Más abajo explicamos qué pasos debe seguir. Para información más detallada consulte el manual del usuario de LicenseServer en el sitio web de Altova.

3.1.5.1 Iniciar LicenseServer, RaptorXML Server

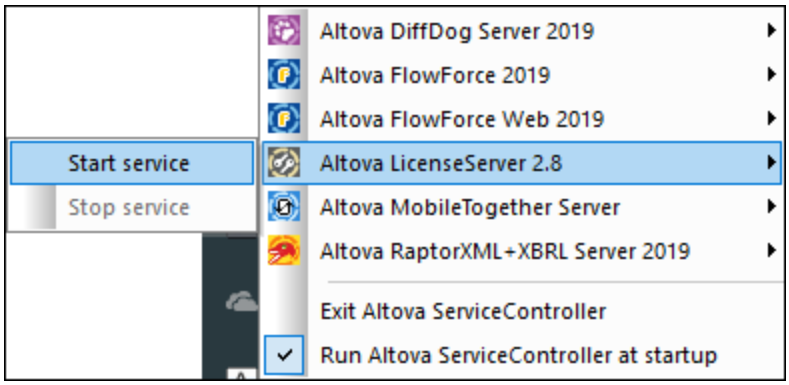
Tanto Altova LicenseServer (LicenseServer para abreviar) como RaptorXML Server se inician con Altova ServiceController.

Altova ServiceController

Altova ServiceController (en adelante *ServiceController*) es una práctica aplicación que sirve para iniciar, detener y configurar los servicios de Altova **en sistemas Windows**.

ServiceController se instala con Altova LicenseServer y con los productos servidor de Altova que se instalan como servicios (DiffDog Server, FlowForce Server, Mobile Together Server y RaptorXML(+XBRL) Server). Una vez iniciado, podrá acceder a ServiceController desde la bandeja del sistema (*imagen siguiente*).

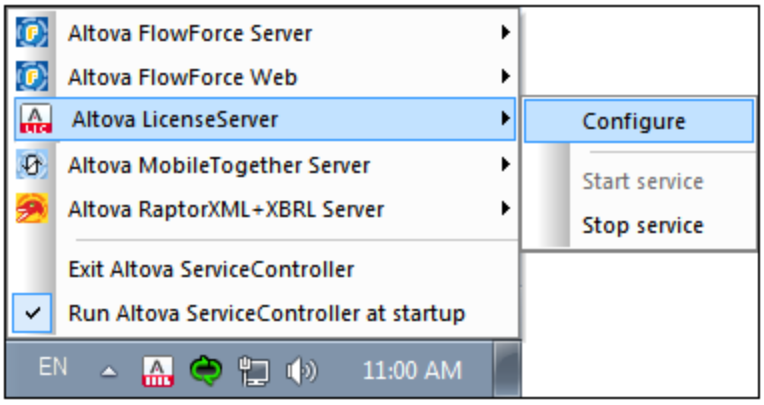
Si quiere que ServiceController se inicie automáticamente nada más iniciar sesión en el sistema, haga clic en el icono de **ServiceController** de la bandeja del sistema para abrir el menú de opciones de **ServiceController** (*imagen siguiente*) y active la opción **Run Altova ServiceController at Startup** (*Ejecutar Altova ServiceController al inicio*), que de todas maneras es la opción predeterminada. Para cerrar ServiceController haga clic en el icono de **ServiceController** de la bandeja del sistema y en el menú haga clic en la opción **Exit Altova ServiceController** (*Salir de Altova ServiceController*).



Iniciar LicenseServer

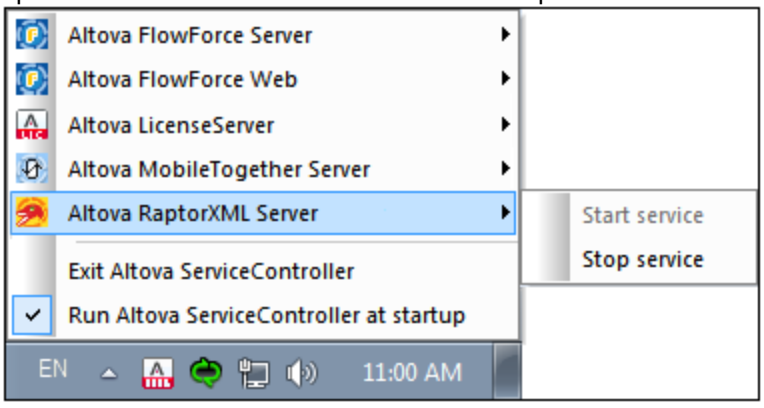
Para iniciar LicenseServer haga clic en el icono de **ServiceController** en la bandeja del sistema, pase el puntero del ratón por encima de Altova LicenseServer en el menú que aparece (*imagen siguiente*) y seleccione **Iniciar servicio** en el submenú de LicenseServer. Si LicenseServer ya está en ejecución, la opción *Start Service* se deshabilita. También puede detener el servicio con ServiceController.

 © 2017-2023 Altova GmbH



Start RaptorXML Server

Para iniciar RaptorXML Server haga clic en el icono de **ServiceController** en la bandeja del sistema, pase el puntero del ratón por encima de Altova RaptorXML Server en el menú que aparece (*imagen siguiente*) y seleccione **Iniciar servicio** en el submenú de RaptorXML Server. Si RaptorXML Server ya está en ejecución, la opción Iniciar servicio se deshabilita. También puede detener el servicio con ServiceController.



Nota: si RaptorXML Server tiene una licencia para ejecutar solamente ejecuciones de subprocesamiento simple (esto suele deberse a que su equipo tiene varios núcleos, pero su licencia es para un solo núcleo), entonces solamente podrá usar una instancia de RaptorXML Server a la vez: bien como servicio, bien desde la línea de comandos. El motivo es que las licencias para un solo núcleo se asignan automáticamente a la primera instancia que se inicia y que está en ejecución. No podrá iniciar una segunda instancia hasta que la primera deje de ejecutarse.

- Si desea usar RaptorXML Server desde la línea de comandos pero el servicio ya está en ejecución, detenga el servicio antes de usar la línea de comandos.
- Si desea iniciar RaptorXML Server como servicio, compruebe que no se está ejecutando ninguna acción de la línea de comandos. Si es así no podrá iniciar el servicio.

3.1.5.2 Registrar RaptorXML Server

Para poder asignar una licencia a RaptorXML Server con Altova LicenseServer debe registrar primero la aplicación con LicenseServer.

Para registrar RaptorXML Server desde la línea de comandos use el comando `licenseserver` e introduzca la dirección del equipo en el que está instalado Licenseserver.

```
RaptorXML licenseserver [opciones] Servidor-O-Dirección-IP
```

Por ejemplo, si el nombre del servidor donde está instalado LicenseServer es `localhost`, entonces debe usar este comando:

```
RaptorXML licenseserver localhost
```

Si RaptorXML Server se instaló como parte de la instalación de FlowForce Server, al registrar FlowForce Server con con LicenseServer también se registrará automáticamente RaptorXML Server. En resumen: (i) Inicie Altova FlowForce Web con ServiceController (*véase punto anterior*), (ii) introduzca su contraseña para acceder a la página de configuración, (iii) seleccione el nombre o la dirección del LicenseServer y haga clic en **Registrar con LicenseServer**. Para más información consulte Registrar FlowForce Server.

Una vez haya registrado su producto correctamente vaya a la pestaña Client Management de la página de configuración de LicenseServer y asigne una licencia a RaptorXML Server.

Nota: para más información sobre cómo registrar los productos de Altova con LicenseServer consulte el manual del usuario de LicenseServer.

3.1.5.3 Asignar una licencia a RaptorXML Server

Una vez haya registrado RaptorXML Server correctamente, la aplicación aparecerá en la pestaña Client Management de la página de configuración de LicenseServer. Ahora puede asignar una licencia a RaptorXML Server.

La asignación de licencias a productos servidor de Altova depende de cuántos núcleos físicos (en contraposición a núcleos lógicos) tiene el procesador del equipo donde se ejecuta el producto servidor de Altova. Por ejemplo, un procesador dual tiene dos núcleos, un procesador *quad* tiene cuatro núcleos, un procesador *hexa-core* tiene seis núcleos, y así sucesivamente. El número de núcleos de la licencia asignada a un producto debe ser mayor o igual al número de núcleos disponibles en dicho equipo servidor, ya sea un servidor físico o un equipo virtual.

Por ejemplo, si un servidor tiene ocho núcleos (un procesador *octa-core*), deberá comprar una licencia para ocho núcleos. También puede combinar varias licencias para alcanzar el número de núcleos necesario. Es decir, puede usar dos licencias para cuatro núcleos para un servidor *octa-core* en lugar de una licencia para ocho núcleos, por ejemplo.

Si usa un equipo servidor con gran cantidad de núcleos, pero tiene un bajo volumen de procesamiento, también puede crear un equipo virtual que tenga adjudicados menos núcleos y comprar una licencia para ese menor número de núcleos. No obstante, dicha implementación sería menos rápida que si utilizaran todos los núcleos disponibles en el servidor.

Nota: cada licencia de los productos servidor de Altova se puede usar de forma simultánea en un equipo

 © 2017-2023 Altova GmbH

como máximo (en el equipo donde está instalado el producto servidor de Altova), incluso si la capacidad de la licencia no está agotada. Por ejemplo, si utiliza una licencia para 10 núcleos para un equipo cliente que tiene 6 núcleos, los 4 núcleos restantes de la licencia no se pueden usar simultáneamente en otro equipo cliente.

Ejecución por subprocesos simples
Si su producto de Altova admite la ejecución por subprocesos simples verá que hay disponible la opción correspondiente. En estos casos, si en el repertorio de licencias hay una licencia de producto servidor de Altova para un solo núcleo, puede asignársela a un equipo que tenga varios núcleos. En este caso, el equipo ejecutará el producto en un solo núcleo. El procesamiento será lógicamente más lento porque solo se usa un núcleo. Es decir, el producto se ejecutará en modo de subprocesamiento simple.

Para asignar una licencia de un solo núcleo a un equipo con varios núcleos basta con marcar en LicenseServer la casilla *Limit to single thread execution* del producto.

Estimación del número de núcleos
Existen varios factores externos que suelen influir en los volúmenes y tiempos de procesamiento que su servidor puede manejar (por ejemplo, el hardware, la carga actual de la CPU, la memoria asignada a otras aplicaciones que se estén ejecutando en el servidor). Para poder conseguir un cálculo lo más exacto posible, recomendamos que primero ejecute las herramientas en su entorno para exponerlas a los factores y datos reales concretos de su negocio.

3.2 Instalación y configuración en Linux

Esta sección explica cómo instalar[34] RaptorXML Server y asignar licencias[37] en sistemas Linux (Debian, Ubuntu, CentOS, RedHat).

Requisitos del sistema (Linux)

- Red Hat Enterprise Linux 7 o superior
- CentOS 7, CentOS Stream 8
- Debian 9 o superior
- Ubuntu 18.04, 20.04, 22.04
- AlmaLinux 9.0
- Rocky Linux 9.0

Requisitos previos

- Realizar la instalación como usuario **root** o como usuario con privilegios **sudo**.
- Debe desinstalar la versión anterior de RaptorXML Server antes de instalar la nueva.
- Si tiene pensado usar la función de gráficos de Altova debe tener instalada al menos una fuente en su sistema para garantizar que los gráficos se visualizan correctamente. Para ver qué fuentes tiene instaladas puede usar el comando **`fc-list`** de la biblioteca Fonconfig.
- Las bibliotecas que aparecen a continuación son un requisito previo para la instalación y ejecución de la aplicación. Si los paquetes que aparecen en esta tabla no están en su equipo Linux, ejecute el comando `yum` (o `apt-get` si procede) para instalarlos.

Requisito para	CentOS, RedHat	Debian	Ubuntu
LicenseServer	libidn o libidn2, krb5-libs, fontconfig	libidn2-0, libgssapi-krb5-2, fontconfig	libidn2-0, libgssapi-krb5-2, libfontconfig
RaptorXML Server	libidn o libidn2, krb5-libs	libidn2-0, libgssapi-krb5-2	libidn2-0, libgssapi-krb5-2

3.2.1 Instalación en Linux

RaptorXML Server se puede instalar en sistemas Linux. A continuación explicamos cómo instalar y configurar su producto. Puede instalarlo como usuario raíz o como usuario con privilegios sudo.

Integración con FlowForce

Si instala RaptorXML Server junto con FlowForce Server, recomendamos que instale primero FlowForce Server. Si instala RaptorXML Server antes que FlowForce Server, una vez haya instalado las dos aplicaciones ejecute este comando:

```
cp /opt/Altova/RaptorXMLServer2024/etc/*.tool /opt/Altova/FlowForceServer2024/tools
```

Con este comando se copia el archivo **.tools** del directorio **/etc** de RaptorXML Server en el directorio **/tools** de FlowForce Server. El archivo **.tool** es necesario para usar FlowForce Server. Este archivo contiene la ruta al

 © 2017-2023 Altova GmbH

ejecutable de RaptorXML Server. Si instala FlowForce antes de instalar RaptorXML Server no necesita ejecutar este comando.

Desinstalar RaptorXML Server

Si tiene que desinstalar una versión anterior de RaptorXML Server siga los pasos que describimos a continuación. En la interfaz de la línea de comandos de Linux, compruebe qué productos servidor de Altova están ya instalados con este comando:

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

Si RaptorXML Server no está instalado, instale el producto tal y como se explica en *Instalar RaptorXML Server*..

Si tiene que desinstalar una versión anterior de RaptorXML Server puede usar este comando:

```
[Debian, Ubuntu]:   sudo dpkg --remove raptorxml
CentOS 7, RedHat 7   sudo rpm -e raptorxml
```

En sistemas Debian y Ubuntu puede ocurrir que RaptorXML Server siga apareciendo en la lista de productos instalados incluso después de haberlo desinstalado. En este caso puede ejecutar el comando **purge** para eliminar RaptorXML Server de la lista. También puede usar el comando **purge** en lugar del comando *instead* que mencionamos más arriba.

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

Descargar el paquete de instalación de RaptorXML Server para Linux

Puede descargar los paquetes de instalación de RaptorXML Server para los sistemas Linux siguientes en el sitio web de Altova.

Distribución	Extensión del paquete
Debian	.deb
Ubuntu	.deb
CentOS	.rpm
RedHat	.rpm

Una vez haya descargado el paquete de instalación para Linux, copie el paquete en cualquier directorio del sistema Linux. Como necesitará un Altova LicenseServer para ejecutar RaptorXML Server, puede que quiera descargar LicenseServer del sitio web de Altova al mismo tiempo que RaptorXML Server en lugar de descargarlo más tarde.

Instalar RaptorXML Server

En una ventana de la terminal cambie al directorio donde copió el paquete Linux. Por ejemplo, si lo copió en un directorio del usuario llamado `MiAltova` (en `/home/User` por ejemplo), cambie a ese directorio con esta línea de comandos:

```
cd /home/User/MiAltova
```

Instale LicenseServer con este comando:

```
[Debian]:   sudo dpkg --install raptorxml-2024-debian.deb
```

```
[Ubuntu]:   sudo dpkg --install raptorxml-2024-ubuntu.deb
[CentOS]:   sudo rpm -ivh raptorxml-2024-1.x86_64.rpm
[RedHat]:   sudo rpm -ivh raptorxml-2024-1.x86_64.rpm
```

Nota: quizás sea necesario ajustar el nombre del paquete anterior para que tenga el número de versión o de service pack actual

El paquete de RaptorXML Server se instalará en la carpeta:

```
/opt/Altova/RaptorXMLServer2024
```

3.2.2 Instalar LicenseServer (Linux)

Para poder utilizar RaptorXML Server debe asignarle una licencia con un Altova LicenseServer de su red. En sistemas Linux deberá instalar Altova LicenseServer por separado. Descargue el instalador de LicenseServer del sitio web de Altova y copie el paquete en cualquier directorio del sistema Linux. Instálelo tal y como instaló RaptorXML Server (consulte *el apartado anterior* 34).

```
[Debian]:   sudo dpkg --install licenseserver-3.12-debian.deb
[Ubuntu]:   sudo dpkg --install licenseserver-3.12-ubuntu.deb
[CentOS]:   sudo rpm -ivh licenseserver-3.12-1.x86_64.rpm
[RedHat]:   sudo rpm -ivh licenseserver-3.12-1.x86_64.rpm
```

El paquete de LicenseServer se instalará en:

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

Para más información sobre cómo registrar RaptorXML Server y asignarle una licencia con Altova LicenseServer consulte el apartado Licensing RaptorXML Server 37. Consulte también LicenseServer documentation para información más detallada.

Versiones de LicenseServer

- Los productos servidor de Altova deben tener una licencia con la versión de LicenseServer correspondiente a la versión de RaptorXML Server instalada o con una versión posterior de LicenseServer.
- La versión de LicenseServer correspondiente a la versión de RaptorXML Server aparece en pantalla durante la instalación de RaptorXML Server. Puede instalar esta versión de LicenseServer junto con RaptorXML Server o puede instalar LicenseServer por separado.
- Antes de instalar una versión nueva de LicenseServer, es necesario desinstalar versiones anteriores. El programa de instalación de LicenseServer se encarga de esto automáticamente si detecta versiones más recientes en el sistema.
- Las versiones de LicenseServer son compatibles y funcionan con versiones más antiguas de RaptorXML Server.
- Si instala una versión nueva de RaptorXML Server y la versión de LicenseServer que está instalada es anterior a la que le corresponde, instale la versión más reciente que está siempre disponible en el sitio web de Altova.
- Cuando se desinstala LicenseServer, todos los datos de registro y asignación de licencias almacenados en la versión antigua de LicenseServer se guardan en una base de datos en el equipo servidor. Estos datos se importan de forma automática a la siguiente versión que se instale en el equipo.
- El número de versión de LicenseServer siempre aparece al final de la página de configuración de

 © 2017-2023 Altova GmbH

LicenseServer.

Versión actual: 3.12

3.2.3 Asignar licencias a RaptorXML Server (Linux)

Para poder trabajar con RaptorXML Server es necesario asignarle una licencia con Altova LicenseServer. La asignación de licencias es un proceso de dos pasos:

1. El primero consiste en **registrar RaptorXML Server** con LicenseServer desde RaptorXML Server.
2. El segundo paso consiste en **asignar una licencia** a RaptorXML Server desde LicenseServer. Descargue la versión más reciente de LicenseServer desde el sitio web de Altova e instálelo en su equipo local o en uno de los equipos de su red.

Más abajo explicamos qué pasos debe seguir. Para información más detallada consulte el manual del usuario de LicenseServer en el sitio web de Altova.

3.2.3.1 Iniciar LicenseServer, RaptorXML Server

En este apartado se describe cómo iniciar Altova LicenseServer (LicenseServer para abreviar) y RaptorXML Server. Debe tener privilegios de administrador (root) para poder iniciar estos programas y ejecutar los comandos como usuario raíz. Si ha iniciado sesión como `root` puede omitir la palabra clave `sudo` de los comandos que describimos a continuación.

Iniciar LicenseServer

Para registrar y asignar una licencia a RaptorXML Server con LicenseServer debe ejecutar LicenseServer como demonio. Para ello use este comando:

[≥ Debian 8], [≥ CentOS 7], [≥ Ubuntu 15]	**sudo systemctl start licenseserver**

Si necesita detener LicenseServer, reemplace **`start`** con **`stop`** en el comando anterior. Por ejemplo:

```
sudo systemctl stop licenseserver
```

Start RaptorXML Server

Inicie RaptorXML Server como demonio con el comando que corresponda a su sistema:

[≥ Debian 8], [≥ CentOS 7], [≥ Ubuntu 15]	**sudo systemctl start raptorxmlserver**

Si necesita detener RaptorXML Server, reemplace start con **`stop`** en el comando anterior. Por ejemplo:

```
sudo systemctl stop raptorxmlserver
```

Comprobar el estado de los demonios

Para comprobar si un demonio de está ejecutando use este comando pero reemplace **servicename** con el nombre del demonio que quiere comprobar:

```
sudo service servicename status
```

3.2.3.2 Registrar RaptorXML Server

Para registrar RaptorXML Server desde la línea de comandos use el comando `licenseserver`:

sudo /opt/Altova/RaptorXMLServer2024/bin/raptorxml licenseserver [opciones] ***Servidor-O-Dirección-IP***

Por ejemplo, si el nombre del servidor donde está instalado LicenseServer es `localhost`, entonces debe usar este comando:

```
sudo /opt/Altova/RaptorXMLServer2024/bin/raptorxml licenseserver localhost
```

En el comando anterior, `localhost` es el nombre del servidor en el que está instalado LicenseServer. Observe que el ejecutable de RaptorXML Server se encuentra en:

/opt/Altova/RaptorXMLServer2024/bin/

Una vez haya registrado su producto correctamente vaya a la pestaña Client Management de la página de configuración de LicenseServer y asigne una licencia a RaptorXML Server.

Nota: para más información sobre cómo registrar los productos de Altova con LicenseServer consulte el manual del usuario de LicenseServer.

3.2.3.3 Asignar licencias a RaptorXML Server

Una vez haya registrado RaptorXML Server correctamente, la aplicación aparecerá en la pestaña Client Management de la página de configuración de LicenseServer. Ahora puede asignar una licencia a RaptorXML Server.

La asignación de licencias a productos servidor de Altova depende de cuántos núcleos físicos (en contraposición a núcleos lógicos) tiene el procesador del equipo donde se ejecuta el producto servidor de Altova. Por ejemplo, un procesador dual tiene dos núcleos, un procesador *quad* tiene cuatro núcleos, un procesador *hexa-core* tiene seis núcleos, y así sucesivamente. El número de núcleos de la licencia asignada a un producto debe ser mayor o igual al número de núcleos disponibles en dicho equipo servidor, ya sea un servidor físico o un equipo virtual.

Por ejemplo, si un servidor tiene ocho núcleos (un procesador *octa-core*), deberá comprar una licencia para ocho núcleos. También puede combinar varias licencias para alcanzar el número de núcleos necesario. Es decir, puede usar dos licencias para cuatro núcleos para un servidor *octa-core* en lugar de una licencia para ocho núcleos, por ejemplo.

Si usa un equipo servidor con gran cantidad de núcleos, pero tiene un bajo volumen de procesamiento, también puede crear un equipo virtual que tenga adjudicados menos núcleos y comprar una licencia para ese menor

 © 2017-2023 Altova GmbH

número de núcleos. No obstante, dicha implementación sería menos rápida que si utilizaran todos los núcleos disponibles en el servidor.

Nota: cada licencia de los productos servidor de Altova se puede usar de forma simultánea en un equipo como máximo (en el equipo donde está instalado el producto servidor de Altova), incluso si la capacidad de la licencia no está agotada. Por ejemplo, si utiliza una licencia para 10 núcleos para un equipo cliente que tiene 6 núcleos, los 4 núcleos restantes de la licencia no se pueden usar simultáneamente en otro equipo cliente.

Ejecución por subprocesos simples
Si su producto de Altova admite la ejecución por subprocesos simples verá que hay disponible la opción correspondiente. En estos casos, si en el repertorio de licencias hay una licencia de producto servidor de Altova para un solo núcleo, puede asignársela a un equipo que tenga varios núcleos. En este caso, el equipo ejecutará el producto en un solo núcleo. El procesamiento será lógicamente más lento porque solo se usa un núcleo. Es decir, el producto se ejecutará en modo de subprocesamiento simple.

Para asignar una licencia de un solo núcleo a un equipo con varios núcleos basta con marcar en LicenseServer la casilla *Limit to single thread execution* del producto.

Estimación del número de núcleos
Existen varios factores externos que suelen influir en los volúmenes y tiempos de procesamiento que su servidor puede manejar (por ejemplo, el hardware, la carga actual de la CPU, la memoria asignada a otras aplicaciones que se estén ejecutando en el servidor). Para poder conseguir un cálculo lo más exacto posible, recomendamos que primero ejecute las herramientas en su entorno para exponerlas a los factores y datos reales concretos de su negocio.

3.3 Instalación y configuración en macOS

Esta sección explica cómo instalar[40] RaptorXML Server y asignarle licencias[42] en sistemas macOS.

Requisitos del sistema (macOS)

- macOS 11 o superior

Requisitos

- Asegúrese de que Altova LicenseServer está instalado y ejecutándose
- Puede instalarlo como usuario raíz o como usuario con privilegios **sudo**.
- Debe desinstalar la versión anterior de RaptorXML Server antes de instalar una nueva.
- si quiere usar la función de gráficos de Altova, entonces debe haber al menos una fuente instalada en su sistema para que los gráficos se muestren correctamente. Para ver una lista de las fuentes instaladas use, por ejemplo, el comando **`fc-list`** de la biblioteca Fontconfig.
- El equipo macOS debe estar configurado de forma que su nombre se resuelva en una dirección de IP. Esto significa que debe poder hacerle ping al nombre de host desde la terminal con el comando **`ping <hostname>`**.

3.3.1 Instalación en macOS

RaptorXML Server se puede instalar en sistemas macOS. A continuación explicamos cómo instalar y configurar su producto.

Integración con FlowForce

Si instala RaptorXML Server junto con FlowForce Server, recomendamos que instale primero FlowForce Server. Si instala RaptorXML Server antes que FlowForce Server, una vez haya instalado las dos aplicaciones ejecute este comando:

```
cp /usr/local/Altova/RaptorXMLServer2024/etc/*.tool /usr/local/Altova/FlowForceServer2024/tools
```

Con este comando se copia el archivo **`.tools`** del directorio **`/etc`** de RaptorXML Server en el directorio **`/tools`** de FlowForce Server. El archivo **`.tool`** es necesario para usar FlowForce Server. Este archivo contiene la ruta al ejecutable de RaptorXML Server. Si instala FlowForce antes de instalar RaptorXML Server no necesita ejecutar este comando.

Desinstalar RaptorXML Server

Antes de desinstalar RaptorXML Server debe detener el servicio con este comando:

```
sudo launchctl unload /Library/LaunchDaemons/com.altova.RaptorXMLServer2024.plist
```

Para comprobar s el servicio se ha detenido abra el Monitor de actividad en Finder y asegúrese de que RaptorXML Server no está en la lista.En la carpeta de aplicaciones de Finder, haga clic con el botón derecho en el icono de RaptorXML Server y seleccione **Mover a la papelera**. La aplicación se envía a la papelera, pero debe quitar la aplicación de la carpeta `usr`. Para ello use el comando:

© 2017-2023 Altova GmbH

```
sudo rm -rf /usr/local/Altova/RaptorXMLServer2024/
```

Si tiene que desinstalar una versión antigua de Altova License Server primer tiene que detener su ejecución como servicio. Para ello use el comando:

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

Para comprobar si el servicio se detuvo o no, abra la terminal del Monitor de actividad en Finder y confirme que LicenseServer no está en la lista. Ahora procesa a desinstalar la aplicación con los mismos pasos que hemos explicado antes para RaptorXML Server.

Instalar RaptorXML Server

1. Descargue el archivo de imagen de disco (`.dmg`) de RaptorXML Server desde el sitio web de Altova (http://www.altova.com/es/download.html).
2. Haga clic en el archivo para abrir la imagen de disco (`.dmg`). El instalador de RaptorXML Server aparece en el equipo como otra unidad de disco más.
3. En el disco virtual haga doble clic en el paquete del instalador (`.pkg`).
4. Siga los pasos del asistente de instalación. En uno de los pasos debe aceptar el contrato de licencia para poder seguir con la instalación. *Consulte también* *Asignar licencias a RaptorXML Server* 42.
5. Para expulsar el disco después de la instalación haga clic en él con el botón derecho y seleccione **Expulsar**.

El paquete de RaptorXML Server se instalará en la carpeta:

`/usr/local/Altova/RaptorXMLServer2024` (archivos binarios de la aplicación)
`/var/Altova/RaptorXMLServer` (archivos de datos: BD y registros)

El servidor demonio de RaptorXML Server se inicia automáticamente después de la instalación y de reiniciar el equipo. Siempre puede iniciar RaptorXML Server como demonio con este comando:

```
sudo launchctl load /Library/LaunchDaemons/com.altova.RaptorXMLServer2024.plist
```

3.3.2 Instalar LicenseServer (macOS)

Para poder utilizar RaptorXML Server debe asignarle una licencia con un Altova LicenseServer de su red. Encontrará el paquete de instalación de LicenseServer en el disco virtual que montó en el paso anterior. Para instalar LicenseServer haga doble clic en el paquete del instalador que incluye el disco virtual y siga las instrucciones que aparecen en pantalla. Para poder continuar con la instalación deberá aceptar el contrato de licencia.

También puede descargar e instalar Altova LicenseServer por separado desde el sitio web de Altova (http://www.altova.com/es/download.html).

El paquete de LicenseServer se instalará en la carpeta:

`/usr/local/Altova/LicenseServer`

Para más información sobre cómo registrar RaptorXML Server con Altova LicenseServer y asignarle una licencia consulte Asignar licencias en macOS 42.

Versiones de LicenseServer

- Los productos servidor de Altova deben tener una licencia con la versión de LicenseServer

correspondiente a la versión de RaptorXML Server instalada o con una versión posterior de LicenseServer.
- La versión de LicenseServer correspondiente a la versión de RaptorXML Server aparece en pantalla durante la instalación de RaptorXML Server. Puede instalar esta versión de LicenseServer junto con RaptorXML Server o puede instalar LicenseServer por separado.
- Antes de instalar una versión nueva de LicenseServer, es necesario desinstalar versiones anteriores. El programa de instalación de LicenseServer se encarga de esto automáticamente si detecta versiones más recientes en el sistema.
- Las versiones de LicenseServer son compatibles y funcionan con versiones más antiguas de RaptorXML Server.
- Si instala una versión nueva de RaptorXML Server y la versión de LicenseServer que está instalada es anterior a la que le corresponde, instale la versión más reciente que está siempre disponible en el sitio web de Altova.
- Cuando se desinstala LicenseServer, todos los datos de registro y asignación de licencias almacenados en la versión antigua de LicenseServer se guardan en una base de datos en el equipo servidor. Estos datos se importan de forma automática a la siguiente versión que se instale en el equipo.
- El número de versión de LicenseServer siempre aparece al final de la página de configuración de LicenseServer.

Versión actual: 3.12

3.3.3 Asignar licencias a RaptorXML Server (macOS)

Para poder trabajar con RaptorXML Server es necesario asignarle una licencia con Altova LicenseServer. La asignación de licencias es un proceso de dos pasos:

1. El primero consiste en **registrar RaptorXML Server** con LicenseServer desde RaptorXML Server.
2. El segundo paso consiste en **asignar una licencia** a RaptorXML Server desde LicenseServer. Descargue la versión más reciente de LicenseServer desde el sitio web de Altova e instálelo en su equipo local o en uno de los equipos de su red.

Más abajo explicamos qué pasos debe seguir. Para información más detallada consulte el manual del usuario de LicenseServer en el sitio web de Altova.

3.3.3.1 Iniciar LicenseServer, RaptorXML Server

En este apartado se describe cómo iniciar Altova LicenseServer (LicenseServer para abreviar) y RaptorXML Server. Debe tener privilegios de administrador (root) para poder iniciar estos programas y ejecutar los comandos como usuario raíz. Si ha iniciado sesión como `root` puede omitir la palabra clave `sudo` de los comandos que describimos a continuación.

Iniciar LicenseServer

Para registrar y asignar una licencia a RaptorXML Server con LicenseServer debe ejecutar LicenseServer como demonio. Para ello use este comando:

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


© 2017-2023 Altova GmbH

Si necesita detener LicenseServer, reemplace **load** con **unload** en el comando anterior.

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

Iniciar RaptorXML Server

El servidor demonio de RaptorXML Server se inicia automáticamente después de la instalación y de reiniciar el equipo. Siempre puede iniciar RaptorXML Server como demonio con este comando:

```
sudo launchctl load /Library/LaunchDaemons/com.altova.RaptorXMLServer2024.plist
```

Si necesita detener RaptorXML Server, utilice este otro comando:

```
sudo launchctl unload /Library/LaunchDaemons/com.altova.RaptorXMLServer2024.plist
```

3.3.3.2 Registrar RaptorXML Server

Para registrar RaptorXML Server desde la línea de comandos use el comando `licenseserver`:

```
sudo /usr/local/Altova/RaptorXMLServer2024/bin/RaptorXML licenseserver [opciones]
Servidor-O-Dirección-IP
```

Por ejemplo, si el nombre del servidor donde está instalado LicenseServer es `localhost`, entonces debe usar este comando:

```
sudo /usr/local/Altova/RaptorXMLServer2024/bin/RaptorXML licenseserver localhost
```

En el comando anterior, `localhost` es el nombre del servidor en el que está instalado LicenseServer. Observe que el ejecutable de RaptorXML Server se encuentra en:

```
/usr/local/Altova/RaptorXMLServer2024/bin/
```

Una vez haya registrado su producto correctamente vaya a la pestaña Client Management de la página de configuración de LicenseServer y asigne una licencia a RaptorXML Server.

Nota: para más información sobre cómo registrar los productos de Altova con LicenseServer consulte el manual del usuario de LicenseServer.

3.3.3.3 Asignar licencias a RaptorXML Server

Una vez haya registrado RaptorXML Server correctamente, la aplicación aparecerá en la pestaña Client Management de la página de configuración de LicenseServer. Ahora puede asignar una licencia a RaptorXML Server.

La asignación de licencias a productos servidor de Altova depende de cuántos núcleos físicos (en contraposición a núcleos lógicos) tiene el procesador del equipo donde se ejecuta el producto servidor de Altova. Por ejemplo, un procesador dual tiene dos núcleos, un procesador *quad* tiene cuatro núcleos, un procesador *hexa-core* tiene seis núcleos, y así sucesivamente. El número de núcleos de la licencia asignada a un producto debe ser mayor o igual al número de núcleos disponibles en dicho equipo servidor, ya sea un servidor físico o un equipo virtual.

Por ejemplo, si un servidor tiene ocho núcleos (un procesador *octa-core*), deberá comprar una licencia para ocho núcleos. También puede combinar varias licencias para alcanzar el número de núcleos necesario. Es decir, puede usar dos licencias para cuatro núcleos para un servidor *octa-core* en lugar de una licencia para

ocho núcleos, por ejemplo.

Si usa un equipo servidor con gran cantidad de núcleos, pero tiene un bajo volumen de procesamiento, también puede crear un equipo virtual que tenga adjudicados menos núcleos y comprar una licencia para ese menor número de núcleos. No obstante, dicha implementación sería menos rápida que si utilizaran todos los núcleos disponibles en el servidor.

Nota: cada licencia de los productos servidor de Altova se puede usar de forma simultánea en un equipo como máximo (en el equipo donde está instalado el producto servidor de Altova), incluso si la capacidad de la licencia no está agotada. Por ejemplo, si utiliza una licencia para 10 núcleos para un equipo cliente que tiene 6 núcleos, los 4 núcleos restantes de la licencia no se pueden usar simultáneamente en otro equipo cliente.

Ejecución por subprocesos simples
Si su producto de Altova admite la ejecución por subprocesos simples verá que hay disponible la opción correspondiente. En estos casos, si en el repertorio de licencias hay una licencia de producto servidor de Altova para un solo núcleo, puede asignársela a un equipo que tenga varios núcleos. En este caso, el equipo ejecutará el producto en un solo núcleo. El procesamiento será lógicamente más lento porque solo se usa un núcleo. Es decir, el producto se ejecutará en modo de subprocesamiento simple.

Para asignar una licencia de un solo núcleo a un equipo con varios núcleos basta con marcar en LicenseServer la casilla *Limit to single thread execution* del producto.

Estimación del número de núcleos
Existen varios factores externos que suelen influir en los volúmenes y tiempos de procesamiento que su servidor puede manejar (por ejemplo, el hardware, la carga actual de la CPU, la memoria asignada a otras aplicaciones que se estén ejecutando en el servidor). Para poder conseguir un cálculo lo más exacto posible, recomendamos que primero ejecute las herramientas en su entorno para exponerlas a los factores y datos reales concretos de su negocio.

© 2017-2023 Altova GmbH

3.4 Actualizar RaptorXML Server

Al actualizar su producto a una versión más reciente de RaptorXML Server puede usar la licencia que ya tiene para esta versión nueva si durante la instalación:

- registra la versión nueva con el mismo LicenseServer con el que registró la versión anterior de RaptorXML Server
- acepta el acuerdo de la licencia de RaptorXML Server.

La forma más fácil de transferir una licencia de una versión anterior de RaptorXML Server a la más reciente es seguir los pasos del proceso de instalación. Estos son los pasos, en el orden en que se llevan a cabo:

1. Deje que el instalador registre la versión nueva de RaptorXML Server con el LicenseServer en el que está la licencia que usaba la versión anterior de RaptorXML Server.
2. Acepte el acuerdo de la licencia de RaptorXML Server. (Si no lo hace no se instalará la versión más reciente.)

Nota: si no registra RaptorXML Server con el LicenseServer correcto durante el proceso de instalación deberá registrar y asignarle una licencia a RaptorXML Server manualmente con su LicenseServer alternativo.

3.5 Migrar RaptorXML Server a un equipo nuevo

Si quiere migrar RaptorXML Server de un equipo a otro (también entre plataformas compatibles) siga las instrucciones de más abajo.

Migrar RaptorXML Server a otro equipo consiste en asignar la licencia del equipo antiguo al equipo nuevo. Para ello siga estas instrucciones:

1. Instale RaptorXML Server en el equipo nuevo. Si ya instaló el producto como parte de la instalación de FlowForce Server, ignore este paso.
2. En el equipo nuevo registre RaptorXML Server con Altova LicenseServer.
3. En el equipo antiguo asegúrese de que ningún cliente está usando el servidor .
4. Abra la página de administración de Altova LicenseServer. Desactive la licencia de RaptorXML Server en el equipo antiguo y vuelva a asignarla en el equipo nuevo.

Nota: Migre el archivo de configuración del servidor para conservar las configuraciones anteriores.

Nota: Si estaba utilizando catálogos XML en el equipo antiguo, migre estos al nuevo equipo.

© 2017-2023 Altova GmbH

4 Procedimientos generales

RaptorXML tiene opciones especiales que admiten el uso de catálogos XML [48] y de recursos globales de Altova [55], características que mejoran la portabilidad y modularidad del entorno en el que se trabaja. Aproveche estas funciones para mejorar el rendimiento de trabajo en su entorno.

Esta sección incluye varios apartados con información sobre:

- Cómo usar los catálogos XML [48].
- Cómo trabajar con los recursos globales de Altova [55].
- Problemas de seguridad [57] relacionados con RaptorXML y cómo solucionarlos.

4.1 Catálogos XML

El mecanismo de catalogación XML permite recuperar archivos de carpetas locales, lo que aumenta la velocidad de procesamiento, además de mejorar la portabilidad de los documentos, ya que solo es necesario cambiar los URIs de los archivos de catálogo. Consulte el apartado Funcionamiento de los catálogos 48 para más información.

Los productos XML de Altova usan este mecanismo de catalogación para que acceder a archivos de uso habitual, como DTDs o esquemas XML, y cargarlos resulte más rápido. Los usuarios pueden personalizar y ampliar este mecanismo, que se describe con más detalle en los apartados Estructura de los catálogos en RaptorXML Server 50 y Personalizar catálogos 51. El apartado Variables para ubicaciones de sistemas Windows 53 contiene la lista de las variables de Windows para ubicaciones habituales del sistema. Estas variables se pueden usar en los archivos de los catálogos para ubicar carpetas de uso habitual.

Este apartado está organizado en varios subapartados:

- Funcionamiento de los catálogos 48
- Estructura de los catálogos en RaptorXML Server 50
- Personalizar catálogos 51
- Variables para ubicaciones de sistemas Windows 53

Para más información sobre los catálogos consulte la especificación XML Catalogs specification.

Instalar esquemas con el Gestor de esquemas

El Gestor de esquemas 382 permite instalar en pocos pasos esquemas relevantes y configurar los archivos de catálogo para poder acceder correctamente a los esquemas instalados. Consulte la sección Gestor de esquemas 382 para más información.

Si un documento se valida con un esquema que no está instalado pero sí disponible con el Gestor de esquemas 382, este se instala automáticamente. Sin embargo, si el paquete de esquemas que se quiere instalar con el Gestor de esquemas contiene asignaciones de espacios de nombres, la instalación automática no funciona, sino que debe ejecutar Gestor de esquemas, seleccionar qué paquetes quiere instalar y ejecutar la instalación. Si después de instalar los paquetes RaptorXML Server no puede encontrar alguno de los componentes del esquema, reinicie la aplicación y vuelva a intentarlo.

4.1.1 Funcionamiento de los catálogos

Los catálogos se pueden usar para redirigir los esquemas DTD y XML. El concepto tras los mecanismos es el mismo en los dos casos, pero los detalles son distintos; los explicamos a continuación.

DTDs

Los catálogos se suelen usar para redirigir una llamada a un DTD o un URI local. Para ello se asignan, en el archivo de catálogo, identificadores públicos o del sistema al URI local necesario. De este modo, cuando se lee la declaración `DOCTYPE` en un archivo XML, el identificador de sistema o público localiza el recurso local necesario con ayuda de la asignación del archivo de catálogo.

Para los esquemas más utilizados el identificador `PUBLIC` suele estar predefinido y, por tanto, sólo hace falta

 © 2017-2023 Altova GmbH

que el URI del archivo de catálogo asigne el identificador **PUBLIC** a la copia local correcta. Cuando se analiza el documento XML, se lee el identificador **PUBLIC** del documento. Si se encuentra este identificador en un archivo de catálogo, se buscará la URL correspondiente del archivo de catálogo y se leerá el esquema desde esta ubicación. Por ejemplo, imaginemos que abrimos este archivo SVG en RaptorXML Server:

```
<?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>
```

Se busca en el catálogo el identificador **PUBLIC** de este archivo SVG. Imaginemos que el archivo de catálogo contiene esta entrada:

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

En este caso hay una coincidencia del identificador **PUBLIC**. La consulta del SVG DTD se redirige al URL **schemas/svg/svg11.dtd** (que es relativo al archivo de catálogo). Este es un archivo local y se usa como DTD para el archivo SCG. Si en el catálogo no hay una asignación para el identificador **Public**, entonces se usa la URL del documento XML (en el archivo SVG del ejemplo anterior la URL de Internet es: **http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd**).

Esquemas XML

En RaptorXML Server también puede usar catálogos para redireccionar a un **esquema XML**. En el archivo de instancia XML, la referencia al esquema sucederá en el atributo **xsi:schemaLocation** del elemento de documento de nivel superior del documento XML. Por ejemplo:

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

El valor del atributo **xsi:schemaLocation** tiene dos partes: un espacio de nombres (verde) y un URI (resaltado). La parte del espacio de nombres se usa en el catálogo para la asignación con el recurso alternativo. Por ejemplo, esta entrada de catálogo redirige la referencia del esquema anterior a un esquema en una ubicación alternativa.

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

Por lo general, la parte URI del valor del atributo **xsi:schemaLocation** es una ruta a la ubicación real del esquema. Sin embargo, si se hace referencia al esquema a través de un catálogo, no es necesario que la parte URI apunte a un esquema XML real, aunque el esquema debe existir para que el atributo **xsi:schemaLocation** siga siendo válido desde el punto de vista léxico. Por ejemplo, el valor `foo` sería suficiente para que la parte URI del valor del atributo sea válida.

4.1.2 Estructura de los catálogos en RaptorXML Server

Al iniciarse, RaptorXML Server carga un archivo llamado **RootCatalog.xml** (*cuya estructura aparece a continuación*), que contiene una lista de los archivos de catálogo que se buscarán. El usuario puede modificar esta lista y añadir tantos archivos de catálogo como desee, escribiendo cada archivo en un elemento **nextCatalog**. La aplicación busca cada uno de estos archivos de catálogo y sus URI se resuelven de acuerdo con sus asignaciones.

Extracto de 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>
```

El extracto anterior hace referencia a un catálogo personalizado (**CustomCatalog.xml**) y a un conjunto de catálogos que localizan los esquemas usados a menudo (como W3C XML Schemas y el esquema SVG).

- **CustomCatalog.xml** se encuentra enla subcarpeta **etc** de la aplicación RaptorXML Server. Debe crear el archivo a partir de una plantilla llamada **CustomCatalog_template.xml**. su carpeta personal (a la que puede acceder con la variable **%PersonalFolder%**). Es un archivo base en el que el usuario puede crear asignaciones propias. Puede añadir asignaciones a **CustomCatalog.xml** para cualquier esquema que necesite que no aparezca en los archivos de catálogo de la carpeta Common Schemas. Debe usar los elementos compatibles del mecanismo de catálogo OASIS (*véase más abajo*).
- La carpeta Common Schemas Folder (a la que se accede con la variable **%CommonSchemasFolder%**) contiene un conjunto de esquemas de uso habitual. Dentro de cada una de estas carpetas hay un archivo **catalog.xml** que asigna identificadores públicos y/o del sistema a URIs que apuntan a copias locales de los esquemas correspondientes.
- **CoreCatalog.xml** está en la carpeta de la aplicación <%XMLSPY%> y se usa para localizar esquemas y hojas de estilos que usan los procesos específicos de <%XMLSPY%>, como los archivos SPS de StyleVision, que se usan para generar documentos XML para la Vista Authentic de Altova.

Tenga en cuenta que:

- Durante una instalación nueva de la misma versión principal (misma u otra subversión), el archivo de plantilla se sustituye por uno nuevo, pero **CustomCatalog.xml** no cambia.
- Sin embargo, si instala una versión principal nueva sobre una versión principal anterior, la carpeta de la instalación anterior se borra junto con su archivo **CustomCatalog.xml**. Por tanto, si quiere seguir usando el **CustomCatalog.xml** de la versión anterior debe guardarlo en una ubicación segura. Después de instalar la versión principal nueva puede copiar el archivo **CustomCatalog.xml** guardado en la carpeta **etc** de la versión principal nueva y editarlo allí como necesite.

© 2017-2023 Altova GmbH

Variables de ubicación
Las variables que se usan en **RootCatalog.xml** (*véase el extracto más arriba*) tienen estos valores:

%PersonalFolder%	La carpeta personal del usuario, por ejemplo C:\Users\<name>\Documents.
%CommonSchemasFolder%	C:\ProgramData\Altova\Common2024\Schemas
%ApplicationWritableDataFolder%	C:\ProgramData\Altova

Ubicación de los archivos de catálogo y los esquemas
Estas son las ubicaciones de los distintos archivos de los catálogos.

- **RootCatalog.xml**, **CustomCatalog.xml**, y **CoreCatalog.xml** se instalan en la carpeta de aplicación de <%XMLSPY%>.
- Cada archivo **catalog.xml** está en una carpeta de esquema y estas carpetas están dentro de la carpeta común de esquemas.

4.1.3 Personalizar catálogos

Cuando cree entradas en el archivo **CustomCatalog.xml** (o en cualquier otro archivo de catálogo que sea leído por RaptorXML Server), utilice únicamente los elementos que aparecen a continuación de la especificación de catálogos OASIS. En la lista que aparece más adelante explicamos los valores de los atributos de cada elemento. Si desea consultar una descripción más detallada, visite la página de la especificación XML Catalogs. Tenga en cuenta que cada uno de los elementos del subconjunto pueden llevar el atributo **xml:base**, que se usa para especificar el URI base del elemento.

- `<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"/>`

Tenga en cuenta que:

- Cuando no exista un identificador público, como es el caso de casi todas las hojas de estilos, el identificador de sistema se puede asignar directamente a una URL con el elemento **system**.
- Un URI se puede asignar a otro URI con el elemento `uri`.
- Los elementos `rewriteURI` y `rewriteSystem` sirven para volver a escribir la parte inicial de un URI o identificador de sistema respectivamente. Gracias a ello se puede sustituir el principio de la ruta de acceso de un archivo y, por consiguiente, se puede apuntar a otro directorio. Para más información sobre estos elementos, consulte la especificación XML Catalogs.

A partir de su versión de 2014 RaptorXML Server cumple escrupulosamente con la especificación de catálogos XML XML Catalogs specification (OASIS Standard V1.1, 7 October 2005) Esta especificación separa estrictamente las consultas por identificador externo (las realizadas con un ID público o de sistema) de las búsquedas por URI (los URI que no son ID públicos ni de sistema). Por tanto, los URIs de espacios de

nombres deben considerarse simplemente como URIs (no como IDs públicos o del sistema) y deben usarse como búsquedas por URI y no como consultas por identificador externo. En las versiones de RaptorXML Server previas a la de 2014 los URIs de espacios de nombres de esquemas se traducían con asignaciones **<public>**. Sin embargo, a partir de la versión 2014 es necesario utilizar asignaciones **<uri>**.

Antes de la versión v2014:
```
<public publicID="http://www.MyMapping.com/ref"
    uri="file:///C:/MyDocs/Catalog/test.xsd"/>
```
A partir de la versión 2014:
```
<uri name="http://www.MyMapping.com/ref"
    uri="file:///C:/MyDocs/Catalog/test.xsd"/>
```

Cómo encuentra RaptorXML Server un esquema de referencia

Para hacer referencia a un esquema desde un documento XML se usa el atributo **xsi:scemaLocation** (*más abajo*). El valor del atributo **xsi:schemaLocation** tiene dos partes: un espacio de nombres (verde) y un URI (resaltado).

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

La secuencia de pasos que debe seguir para encontrar un esquema al que se hace referencia depende de las opciones de validación **--schemalocation-hints** y **--schema-mapping**. A continuación puede encontrar los procedimientos para cada uno de los valores de estas dos opciones:

- **--schemalocation-hints=load-by-schemalocation | load-by-namespace | load-combining-both | ignore**
  Indica el comportamiento de los atributos **xsi:schemaLocation** y **xsi:noNamespaceSchemaLocation**: si deben cargar un documento de esquema y, en ese caso, qué información deben usar para encontrarlo (la opción predeterminada es **load-by-schemalocation**).
  - **load-by-schemalocation**
    1. Si la parte del URI de **xsi:schemaLocation** está asignada a un catálogo, carga el URI resultante.
    2. Carga el URI directamente.
  - **load-by-namespace**
    1. Si la parte del espacio de nombres de **xsi:schemaLocation** está asignada a un catálogo, carga el URI resultante.
    2. No carga nada.
  - **load-combining-both**
    1. Si la parte del URI de **xsi:schemaLocation** está asignada a un catálogo, carga el URI resultante.
    2. Si la parte del espacio de nombres de **xsi:schemaLocation** está asignada a un catálogo, carga el URI resultante.
    3. Carga la parte del URI directamente.

- **--schema-mapping=prefer-schemalocation | prefer-namespace**
  Si se usan tanto la ubicación como el espacio de nombres del esquema para encontrar un documento de esquema, entonces esta opción indica cuál de estas dos opciones se prefiere al buscar un catálogo (la opción predeterminada es **schemalocation**). Esta opción se usa para cambiar el orden de los primeros dos pasos en la variante **load-combining-both** de más arriba.

© 2017-2023 Altova GmbH

Especificaciones de XML Schema

La información de la especificación XML Schema está integrada en RaptorXML Server y esta información interna se usa para validar los documentos de esquema XML (`.xsd`). Por tanto, en los documentos de esquema XML no se deberían hacer referencia a ningún esquema que defina la especificación XML Schema.

El archivo **`catalog.xml`** de la carpeta `%AltovaCommonSchemasFolder%\Schemas\schema` incluye referencias a las DTD que implementan especificaciones antiguas de XML Schema. Rogamos no valide sus documentos de esquema XML con estos esquemas. Los archivos referenciados se incluyen con el único objetivo de aportar información a RaptorXML Server para sus ayudantes de entrada, en caso de que el usuario quiera crear documentos basados en estas recomendaciones.

4.1.4 Variables para ubicaciones de sistemas Windows

En el elemento `nextCatalog` puede utilizar algunas variables de entorno Shell para indicar la ruta de acceso a las ubicaciones del sistema (*ver el fragmento anterior del archivo RootCatalog.xml*). Estas son las variables de entorno Shell compatibles:

`%PersonalFolder%`	Ruta de acceso completa de la carpeta personal del usuario actual, por ejemplo `C:\Usuarios\<nombre>\Documentos`
`%CommonSchemasFolder%`	`C:\ProgramData\Altova\Common2024\Schemas`
`%ApplicationWritableDataFolder%`	`C:\ProgramData\Altova`
`%AltovaCommonFolder%`	C:\Archivos de programa\Altova\Common2024
`%DesktopFolder%`	Ruta de acceso completa de la carpeta Escritorio del usuario actual.
`%ProgramMenuFolder%`	Ruta de acceso completa de la carpeta del menú Programas del usuario actual.
`%StartMenuFolder%`	Ruta de acceso completa de la carpeta del menú Inicio del usuario actual.
`%StartUpFolder%`	Ruta de acceso completa de la carpeta Inicio del usuario actual.
`%TemplateFolder%`	Ruta de acceso completa de la carpeta de plantillas del usuario actual.
`%AdminToolsFolder%`	Ruta de acceso completa del directorio del sistema de archivos que almacena las herramientas administrativas del usuario actual.
`%AppDataFolder%`	Ruta de acceso completa de la carpeta Datos de programa del usuario actual.
`%CommonAppDataFolder%`	Ruta de acceso completa al directorio del sistema de archivos que sirve como repositorio de datos para aplicaciones locales (no roaming).

%FavoritesFolder%	Ruta de acceso completa de la carpeta Favoritos del usuario actual.
%PersonalFolder%	Ruta de acceso completa de la carpeta personal del usuario actual.
%SendToFolder%	Ruta de acceso completa de la carpeta SendTo del usuario actual.
%FontsFolder%	Ruta de acceso completa de la carpeta Fuentes del sistema.
%ProgramFilesFolder%	Ruta de acceso completa de la carpeta Archivos de programa del usuario actual.
%CommonFilesFolder%	Ruta de acceso completa de la carpeta Common files del usuario actual.
%WindowsFolder%	Ruta de acceso completa de la carpeta Windows del usuario actual.
%SystemFolder%	Ruta de acceso completa de la carpeta System del usuario actual.
%LocalAppDataFolder%	Ruta de acceso completa al directorio del sistema de archivos que sirve como repositorio de datos para aplicaciones locales (no roaming).
%MyPicturesFolder%	Ruta de acceso completa a la carpeta Mis imágenes.

© 2017-2023 Altova GmbH

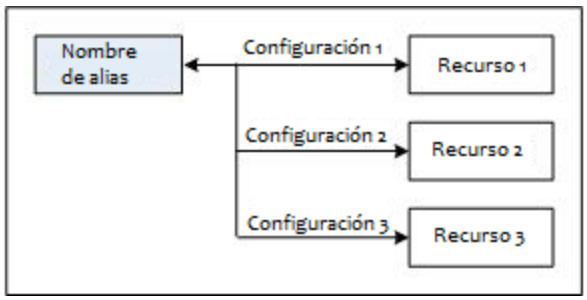
4.2 Recursos globales

Temas de este apartado:

- Recursos globales: ¿qué son? 55
- Recursos globales: ¿cómo se usan? 55

¿Qué son los recursos globales?

Un archivo de recurso global de Altova asigna un alias a varios recursos mediante configuraciones diferentes, tal y como muestra el diagrama que aparece a continuación. La idea es poder cambiar de alias para acceder a recursos distintos, dependiendo de la configuración elegida.



Los recursos globales se definen desde las herramientas de Altova (como Altova XMLSpy, por ejemplo) y se guardan en un archivo XML de recursos globales. RaptorXML puede usar estos recursos globales como datos de entrada. Para ello necesita el nombre y la ubicación del archivo de recursos globales, así como el alias y la configuración que debe usar.

La ventaja de usar recursos globales es que puede cambiar de recurso con solo cambiar el nombre de la configuración. En RaptorXML, esto significa que al usar un valor diferente de la opción `--globalresourcesconfig | --gc`, se puede usar un recurso global distinto (*ver ejemplo que aparece más abajo*).

¿Cómo se utilizan los recursos globales con RaptorXML?

Para especificar el uso de un recurso global como entrada para un comando de RaptorXML es obligatorio usar estos parámetros en la interfaz de la línea de comandos:

- El archivo XML de recursos globales (opción `--globalresourcesfile | --gr`)
- La configuración necesaria (opción `--globalresourcesconfig | --gc`)
- El alias, que se puede especificar directamente en la ILC cuando sea necesario un nombre de archivo. También puede estar dentro del archivo XML en el que RaptorXML busca un nombre de archivo (como en un atributo `xsi:schemaLocation`, por ejemplo).

Por ejemplo, si quiere transformar `entrada.xml` con `transform.xslt` en `salida.html`, lo normal sería usar estos comandos en la ILC usando los nombres de archivo:

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

No obstante, si tiene una definición de recurso global para el alias `MiEntrada` que apunta al recurso de archivo `PrimeraEntrada.xml` por medio de una configuración llamada `PrimeraConfig`, podría usar el alias `MiEntrada` en la línea de comandos:

```
raptorxml xslt --input=altova://file_resource/MiEntrada --gr=C:\MisRecursosGlobales.xml
--gc=PrimeraConfig --output=Salida.html transform.xslt
```

Ahora imagine que tiene otro recurso de archivo, por ejemplo `SegundaEntrada.xml`, que apunta al alias `MiEntrada` por medio de una configuración llamada `SegundaConfig`, entonces puede usar este otro recurso con solo cambiar la opción `--gc` del comando anterior:

```
raptorxml xslt --input=altova://file_resource/MiEntrada --gr=C:\MisRecursosGlobales.xml
--gc=SegundaConfig --output=Salida.html transform.xslt
```

Nota: en el ejemplo anterior se usó un recurso de archivo. Los recursos de archivo deben llevar el prefijo **`altova://file_resource/`**. También puede usar recursos globales que sean carpetas. Para identificar un recurso de carpeta, utilice el prefijo: **`altova://folder_resource/NombreAlias`**. No olvide que en la interfaz de la línea de comandos puede usar recursos de carpeta como parte de la ruta de acceso. Por ejemplo: **`altova://folder_resource/NombreAlias/entrada.xml`**.

 © 2017-2023 Altova GmbH

4.3 Problemas de seguridad

Temas de este apartado:

- Problemas de seguridad relacionados con la interfaz HTTP 57
- Trabajar con scripts Python seguros 57

Algunas características de las interfaces de RaptorXML Server plantean algunos problemas de seguridad, que describimos a continuación junto con soluciones para remediarlos.

Problemas de seguridad relacionados con la interfaz HTTP REST

La interfaz HTTP REST permite por defecto escribir documentos de resultados en cualquier ubicación indicada por el cliente (y a la que se pueda acceder con el protocolo HTTP). Por tanto, es importante tener en cuenta este aspecto de seguridad cuando instale y configure RaptorXML Server.

Si le preocupa que esto pueda comprometer la seguridad de su sistema o que la interfaz se utilice de forma incorrecta, puede configurar el servidor para que escriba los documentos de resultados en un directorio de salida específico del servidor mismo. Esto se consigue estableciendo el valor `false` para la opción `server.unrestricted-filesystem-access` 259 del archivo de configuración del servidor. Si se limita así el acceso, el cliente puede descargar los documentos de resultados del directorio de salida específico mediante solicitudes `GET`. Otra opción es que el administrador copie/cargue los documentos de resultados del servidor en la ubicación de destino.

Trabajar con scripts Python seguros

Cuando se especifica un script Python por HTTP para RaptorXML Server, el script solo funciona si está ubicado en el directorio de confianza 259. El script se ejecuta desde el directorio de confianza. Si especifica un script de cualquier otro directorio, se produce un error. El directorio de confianza se define en la opción `server.script-root-dir` 259 del archivo de configuración del servidor 258 y **es obligatorio** especificar un directorio de confianza si quiere usar scripts Python. Por tanto, asegúrese de guardar en este directorio todos los scripts Python que desea usar.

Aunque todos los resultados generados por el servidor para solicitudes de trabajo HTTP se escriben en el directorio de salida de trabajos 259 (que es un subdirectorio de `output-root-directory` 259), esta limitación no afecta a los scripts Python, que pueden escribir en cualquier ubicación. El administrador del servidor debería revisar los scripts Python del directorio de confianza 259 para evitar problemas de seguridad.

5 Interfaz de la línea de comandos (ILC)

El ejecutable de RaptorXML Server permite usar la línea de comandos (ILC) para llamar a funciones de la aplicación. La ruta de acceso del ejecutable es:

Linux	`/opt/Altova/RaptorXMLServer2024/bin/`**`raptorxml`**
Mac	`/usr/local/Altova/RaptorXMLServer2024/bin/`**`raptorxml`**
Windows	`<`*`CarpetaArchivosPrograma`*`>\Altova\RaptorXMLServer2024\bin\`**`RaptorXML.exe`**

La sintaxis de la línea de comandos es:

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

- `--help` (`--h` en versión corta) muestra el texto de ayuda del comando dado. Si no se indica ningún comando, entonces se enumeran todos los comandos del ejecutable, cada uno con una breve descripción.
- `--version` muestra el número de versión de RaptorXML Server.
- `<command>` es el comando que se ejecuta. Los comandos se describen en las subsecciones de este apartado *(véase la lista más abajo)*.
- `[opciones]` son las opciones de un comando; se enumeran y describen con sus comandos correspondientes.
- `[arguments]` son los argumentos de un comando; se enumeran y describen con sus comandos correspondientes.

▼ Uso de mayúsculas/minúsculas y de barras diagonales en la línea de comandos

`RaptorXML` *en Windows*
`raptorxml` *en Unix (Linux, Mac)*

* Las minúsculas (`raptorxml`) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que la mezcla de mayúsculas y minúsculas (`RaptorXML`) sólo funciona en Windows y Mac.
* En Linux y Mac utilice barras diagonales. En Windows utilice barras diagonales inversas.

Comandos de la ILC

A continuación se describen todos los comandos de la ILC de RaptorXML ordenados según su función.

- Comandos para validar XML, DTD, XSD 59
- Comandos para comprobar el formato 83
- Comandos XQuery 96
- Comandos XSLT 128
- Comandos JSON/Avro 144
- Comandos XML Signature 196
- Comandos generales 209
- Comandos de localización 213
- Comandos de licencias 217
- Comandos de ayuda y licencias 222

 © 2017-2023 Altova GmbH

5.1 Comandos para validar XML, DTD, XSD

Los comandos de validación XML sirven para validar este tipo de documentos:

valxml-withdtd | xml 59	Valida un documento de instancia XML con una DTD.
valxml-withxsd | xsi 63	Valida un documento de instancia XML con un esquema XML.
valdtd | dtd 71	Valida un documento DTD.
valxsd | xsd 75	Valida un documento de esquema XML del W3C (XSD).

5.1.1 valxml-withdtd (xml)

El comando **valxml-withdtd | xml** valida uno o varios documentos XML de instancia con una DTD.

Windows **RaptorXML valxml-withdtd | xml [opciones]** ***ArchivoEntrada***

Linux **raptorxml valxml-withdtd | xml [opciones]** ***ArchivoEntrada***

Mac **raptorxml valxml-withdtd | xml [opciones]** ***ArchivoEntrada***

- El argumento *ArchivoEntrada* es el documento XML que debe validarse. Si existe una referencia a una DTD en el documento, no es necesario usar la opción `--dtd`.
- Si desea validar varios documentos, tiene dos opciones: (i) enumerar los archivos que se deben validar en la línea de comandos, separados por espacios o (ii) enumerar los archivos que se deben validar en un archivo de texto (`.txt`) donde aparece un nombre de archivo por línea y dar este archivo de texto como argumento *ArchivoEntrada* junto con la opción `--listfile` 238 con valor `true` (*ver lista de opciones más abajo*).

Ejemplos

Ejemplos del comando **valxml-withdtd**:

- **raptorxml** valxml-withdtd --dtd=c:\MiDTD.dtd c:\Test.xml
- **raptorxml** xml c:\Test.xml
- **raptorxml** xml --verbose=true c:\Test.xml
- **raptorxml** xml --listfile=true c:\ListaArchivos.txt

▼ Mayúsculas/minúsculas y barras en la línea de comandos

RaptorXML (y **RaptorXMLServer** para comandos administrativos) *en Windows*
raptorxml (y **raptorxmlserver** para comandos administrativos) *en Windows y Unix (Linux, Mac)*

* Las minúsculas (`raptorxml` y `raptorxmlserver`) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que las mayúsculas (`RaptorXML`) funcionan solamente en Windows y Mac.
* Use la barra diagonal en Linux y Mac y la barra diagonal inversa en Windows.

▼ Barra diagonal inversa y espacios en sistemas Windows

En sistemas Windows: si hay espacios o caracteres especiales (por ejemplo en los nombres de archivos o de carpetas, empresas, personas o productos) debe usar comillas: por ejemplo, **"Mi archivo"**. Sin embargo, debe tener en cuenta que una barra diagonal inversa seguida de comillas dobles (por ejemplo, **"C:\Mi Directorio\"**) es posible que no se lea correctamente. Esto se debe a que la barra diagonal inversa también se usa para empezar secuencias de escape y la secuencia de escape **\"** también usa las comillas dobles. Para evitar secuencia de caracteres puede añadir otra barra diagonal inversa: **\\"**. En resumen: si necesita escribir una ruta que contenga espacios y una barra diagonal inversa, esta es la mejor forma de hacerlo: **"C:\Mi Directorio\\"**.

Opciones

Las opciones del comando aparecen más abajo divididas en grupos. Los valores se pueden dar sin comillas excepto en estos dos casos: (i) cuando la cadena de valor contiene espacios y (ii) cuando en la descripción de la opción se indique explícitamente que es necesario el uso de comillas. Si una opción toma un valor booleano y no se indica ningún valor, entonces el valor predeterminado de la opción es `TRUE`. Use la opción **--h, --help** para ver información sobre el comando.

Validación y procesamiento

dtd

--dtd = *ARCHIVO*
Especifica el documento DTD externo que debe utilizarse para la validación. Si en el documento XML hay una referencia a una DTD externa, esta opción de la ILC reemplaza a la referencia externa.

listfile

--listfile = `true|false`
Si el valor es `true`, el argumento *ArchivoEntrada* del comando se entiende como un archivo de texto que contiene un nombre de archivo por línea. Otra opción es enumerar los archivos en la ILC, separados por un espacio. No obstante, recuerde que las ILC tienen un límite de caracteres. Además, no olvide que la opción `--listfile` solamente afecta a los argumentos y no a las opciones.
Valor predeterminado: `false`.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

namespaces

--namespaces = `true|false`
Habilita el procesamiento preparado para espacios de nombres. Esta opción es muy útil si quiere buscar en la instancia XML errores resultantes de espacios de nombres erróneos.
Valor predeterminado: `false`.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

recurse

--recurse = `true|false`
Esta opción se utiliza para seleccionar ficheros dentro de subdirectorios (incluso en archivos ZIP). Si el valor es `true`, el argumento *ArchivoEntrada* del comando seleccionará el fichero seleccionado

 © 2017-2023 Altova GmbH

también en los subdirectorios. Por ejemplo: `"test.zip|zip\test.xml"` seleccionará los ficheros llamados `test.xml` en todos los subdirectorios de la carpeta ZIP. Si quiere puede usar los caracteres comodín `*` y `?`. Por ejemplo: `*.xml` seleccionaría todos los ficheros de la carpeta ZIP que tengan la extensión `.xml`. Valor predeterminado: `false`
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- streaming

  `--streaming =` `true|false`
  Habilita la transmisión por secuencias. En el modo de transmisión por secuencias, el almacenamiento de datos en memoria se reduce al mínimo y el procesamiento es más rápido. El inconveniente es que puede que no esté disponible cierta información que podría necesitar más adelante, como el modelo de datos del documento XML, por ejemplo. Si quiere evitar esto, debería deshabilitar el modo de transmisión por secuencias (dándole el valor `false` a la opción `--streaming`). Cuando use la opción `--script` con el comando `valxml-withxsd`, aconsejamos deshabilitar la transmisión por secuencias. Recuerde que la opción `--streaming` se ignora, si el valor de `--parallel-assessment` es `true`.
  Valor predeterminado: `true`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- Catálogos y recursos globales
  - catalog

    `--catalog =` *ARCHIVO*
    Especifica la ruta de acceso absoluta a un archivo de catálogo que no está en el archivo de catálogo raíz instalado. El valor predeterminado es la ruta de acceso absoluta del archivo de catálogo raíz instalado (`<carpeta-instalación>\Altova\RaptorXMLServer2024\etc\RootCatalog.xml`).
    Consulte el apartado Catálogos XML [48] para obtener más información.

  - user-catalog

    `--user-catalog =` *ARCHIVO*
    Especifica la ruta de acceso absoluta a un catálogo XML que debe utilizarse junto con el catálogo raíz. Consulte el apartado Catálogos XML [48] para obtener más información.

  - enable-globalresources

    `--enable-globalresources =` `true|false`
    Habilita la función de recursos globales [55]. Valor predeterminado: `false`.
    Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

  - globalresourceconfig [gc]

    `--gc | --globalresourceconfig =` *VALOR*
    Especifica la configuración activa del recurso global [55] (y habilita los recursos globales) [55].

  - globalresourcefile [gr]

    `--gr | --globalresourcefile =` *ARCHIVO*

Especifica el archivo de recursos globales[55] (y habilita los recursos globales)[55].

- Opciones comunes
  - error-format

    --error-format = `text|shortxml|longxml`
    Especifica el formato de la salida de error. Los valores posibles son formatos de texto, XML y XML detallado (`longxml`). Valor predeterminado: `text`.

  - error-limit

    --error-limit = *N | unlimited*
    Especifica el límite de errores con un valor comprendido entre `1` y `9999` o `unlimited` (ilimitado). El valor predeterminado es `100`. Cuando se alcanza el límite de error, se detiene la validación. Esta opción es muy práctica a la hora de limitar el uso del procesador durante la validación o transformación.

  - info-limit

    --info-limit = *N | unlimited*
    Indica el límite del mensaje de información dentro del rango `1-65535` or `unlimited`. Si se alcanza el límite de información indicado, el procesamiento continúa pero ya no se informa de más mensajes. El valor predeterminado es `100`.

  - help

    --help
    Muestra el texto de ayuda para el comando. Por ejemplo `valany --h`. (Otra opción es usar el comando `help` con un argumento. Por ejemplo: `help valany`).

  - listfile

    --listfile = `true|false`
    Si el valor es `true`, el argumento *ArchivoEntrada* del comando se entiende como un archivo de texto que contiene un nombre de archivo por línea. Otra opción es enumerar los archivos en la ILC, separados por un espacio. No obstante, recuerde que las ILC tienen un límite de caracteres. Además, no olvide que la opción `--listfile` solamente afecta a los argumentos y no a las opciones.
    Valor predeterminado: `false`.
    Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

  - log-output

    --log-output = *ARCHIVO*
    Escribe el registro de salida en la URL de archivo indicada. Compruebe que la ILC tiene permiso de escritura en la ubicación de destino.

  - network-timeout

    --network-timeout = *VALOR*
    Indica el tiempo de espera en milisegundos para operaciones remotas de entrada y salida. Valor

 © 2017-2023 Altova GmbH

predeterminado: `40000`.

- recurse

  `--recurse =` `true|false`
  Esta opción se utiliza para seleccionar ficheros dentro de subdirectorios (incluso en archivos ZIP). Si el valor es `true`, el argumento *`ArchivoEntrada`* del comando seleccionará el fichero seleccionado también en los subdirectorios. Por ejemplo: `"test.zip|zip\test.xml"` seleccionará los ficheros llamados `test.xml` en todos los subdirectorios de la carpeta ZIP. Si quiere puede usar los caracteres comodín `*` y `?`. Por ejemplo: `*.xml` seleccionaría todos los ficheros de la carpeta ZIP que tengan la extensión `.xml`. Valor predeterminado: `false`
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- verbose

  `--verbose =` `true|false`
  Si el valor es `true`, se genera información adicional durante la validación. Valor predeterminado es `false`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- verbose-output

  `--verbose-output =` *`ARCHIVO`*
  Escribe el resultado detallado en el *`ARCHIVO`* indicado.

- version

  `--version`
  Muestra el número de versión de RaptorXML Server. Si se utiliza con un comando, escriba la opción `--version` antes del comando.

- warning-limit

  `--warning-limit =` *`N | unlimited`*
  Especifica el límite de advertencia en el rango `1-65535` o `unlimited` (ilimitado). El procesamiento continua si se alcanza el límite pero no se registrarán más advertencias. El valor por defecto es `100`.

5.1.2 valxml-withxsd (xsi)

El comando **`valxml-withxsd | xsi`** valida uno o varios documentos XML de instancia con las especificaciones XML Schema Definition Language (XSD) 1.0 y 1.1 del W3C.

Windows **`RaptorXML valxml-withxsd | xsi [opciones] ArchivoEntrada`**

Linux **`raptorxml valxml-withxsd | xsi [opciones] ArchivoEntrada`**

Mac **raptorxml valxml-withxsd | xsi [opciones] *ArchivoEntrada***

- El argumento **ArchivoEntrada** suministra el documento XML que debe validarse. La opción --schemalocation-hints[240] indica qué mecanismo se usa para encontrar el esquema. La opción --xsd=ARCHIVO[239] indica qué esquema se utiliza si en el archivo XML no hay ninguna referencia a un esquema.
- Si desea validar varios documentos, tiene dos opciones: (i) enumerar los archivos que se deben validar en la línea de comandos, separados por espacios o (ii) enumerar los archivos que se deben validar en un archivo de texto (`.txt`) donde aparece un nombre de archivo por línea y dar este archivo de texto como argumento *ArchivoEntrada* junto con la opción --listfile[238] con valor `true` (*ver lista de opciones más abajo*).

Nota: si usa la opción `--script` para ejecutar scripts Python[371], no olvide especificar la opción `--streaming=false`.

Ejemplos

Ejemplos del comando **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

▼ Mayúsculas/minúsculas y barras en la línea de comandos

RaptorXML (y **RaptorXMLServer** para comandos administrativos) *en Windows*
raptorxml (y **raptorxmlserver** para comandos administrativos) *en Windows y Unix (Linux, Mac)*

* Las minúsculas (raptorxml y raptorxmlserver) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que las mayúsculas (RaptorXML) funcionan solamente en Windows y Mac.
* Use la barra diagonal en Linux y Mac y la barra diagonal inversa en Windows.

▼ Barra diagonal inversa y espacios en sistemas Windows

En sistemas Windows: si hay espacios o caracteres especiales (por ejemplo en los nombres de archivos o de carpetas, empresas, personas o productos) debe usar comillas: por ejemplo, **"Mi archivo"**. Sin embargo, debe tener en cuenta que una barra diagonal inversa seguida de comillas dobles (por ejemplo, **"C:\Mi Directorio\"**) es posible que no se lea correctamente. Esto se debe a que la barra diagonal inversa también se usa para empezar secuencias de escape y la secuencia de escape **\"** también usa las comillas dobles. Para evitar secuencia de caracteres puede añadir otra barra diagonal inversa: **\\"**. En resumen: si necesita escribir una ruta que contenga espacios y una barra diagonal inversa, esta es la mejor forma de hacerlo: **"C:\Mi Directorio\\"**.

Opciones

Las opciones del comando aparecen más abajo divididas en grupos. Los valores se pueden dar sin comillas excepto en estos dos casos: (i) cuando la cadena de valor contiene espacios y (ii) cuando en la descripción de la opción se indique explícitamente que es necesario el uso de comillas. Si una opción toma un valor booleano y no se indica ningún valor, entonces el valor predeterminado de la opción es `TRUE`. Use la opción **--h, --help** para ver información sobre el comando.

 © 2017-2023 Altova GmbH

- Validación y procesamiento
  - assessment-mode

    `--assessment-mode =` `lax|strict`
    Especifica el modo de evaluación de la validez del esquema, según se define en las especificaciones XSD. El documento XML de instancia se validará en función del modo especificado en esta opción.
    Valor predeterminado: `strict`.

  - ct-restrict-mode

    `--ct-restrict-mode =` `1.0|1.1|default`
    Especifica cómo comprobar restricciones de tipo complejo. Un valor de **`1.0`** comprueba restricciones de tipo complejo conforme a lo definido en la especificación XSD 1.0 (incluso estando en modo de validación XSD 1.1). Un valor de **`1.1`** comprueba restricciones de tipo complejo conforme a los definido en la especificación XSD 1.1 (incluso estando en modo de validación XSD 1.0). Un valor de **`default`** comprueba restricciones de tipo complejo conforme a lo definido en la especificación XSD del modo de validación habilitado en ese momento (1.0 o 1.1). El valor predeterminado es **`default`**.

  - listfile

    `--listfile =` `true|false`
    Si el valor es `true`, el argumento *`ArchivoEntrada`* del comando se entiende como un archivo de texto que contiene un nombre de archivo por línea. Otra opción es enumerar los archivos en la ILC, separados por un espacio. No obstante, recuerde que las ILC tienen un límite de caracteres. Además, no olvide que la opción `--listfile` solamente afecta a los argumentos y no a las opciones.
    Valor predeterminado: `false`.
    Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

  - parallel-assessment [pa]

    `--pa | --parallel-assessment =` `true|false`
    Si el valor es `true`, la evaluación de la validez de esquemas se realiza en paralelo. Esto significa que si hay más de 128 elementos en cualquiera de los niveles, estos elementos se procesan en paralelo utilizando varios subprocesos. Por tanto, los archivos XML de gran tamaño se pueden procesar más rápido si se habilita esta opción. La evaluación en paralelo se lleva cabo nivel por nivel, pero puede ocurrir en varios niveles de un mismo conjunto de información. Recuerde que la evaluación en paralelo no funciona en modo de transmisión por secuencias. Por este motivo la opción `--streaming` se pasa por alto si el valor de la opción `--parallel-assessment` es `true`. Además, se usa más memoria cuando se utiliza la opción `--parallel-assessment`. El valor predeterminado de esta opción es false y su forma abreviada es `--pa`.
    Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

  - recurse

    `--recurse =` `true|false`
    Esta opción se utiliza para seleccionar ficheros dentro de subdirectorios (incluso en archivos ZIP). Si el valor es `true`, el argumento *`ArchivoEntrada`* del comando seleccionará el fichero seleccionado también en los subdirectorios. Por ejemplo: `"test.zip|zip\test.xml"` seleccionará los ficheros llamados `test.xml` en todos los subdirectorios de la carpeta ZIP. Si quiere puede usar los caracteres comodín `*` y `?`. Por ejemplo: `*.xml` seleccionaría todos los ficheros de la carpeta ZIP que

tengan la extensión `.xml`. Valor predeterminado: `false`
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

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

`--report-import-namespace-mismatch-as-warning =` `true|false`
Informa de los errores de disparidad del espacio de nombres o del espacio de nombres de destino al importar esquemas con **`xs:import`** como advertencias en vez de como errores. El valor predeterminado es `false`.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

▼ schema-imports

`--schema-imports =` `load-by-schemalocation | load-preferring-schemalocation | load-by-namespace | load-combining-both | license-namespace-only`
Esta opción indica el comportamiento de los elementos `xs:import`. Cada uno de estos elementos tiene un atributo opcional `namespace` y un atributo opcional `schemaLocation`: `<import namespace="unEspacioNombres" schemaLocation="unaURL">`. La opción indica si se debe cargar un documento de esquema o solo autorizar a un espacio de nombres. Si la opción indica que se debe cargar un documento de esquema, entonces indica también qué información debe utilizarse para encontrar el documento de esquema. Valor predeterminado: `load-preferring-schemalocation`.

- `load-by-schemalocation`: el valor del atributo `schemaLocation` se utiliza para buscar el esquema, teniendo en cuenta las asignaciones de catálogo[48]. Si está presente el atributo `namespace`, se importa el espacio de nombres (con licencia).
- `load-preferring-schemalocation`: si está presente, se utiliza el atributo `schemaLocation` teniendo en cuenta las asignaciones de catálogo[48]. Si no está presente el atributo `schemaLocation`, entonces se usa el valor del atributo `namespace` a través de las asignaciones de catálogo[48]. Este es el **valor predeterminado**.
- `load-by-namespace`: el valor del atributo `namespace` se utiliza para buscar el esquema por medio de una asignación de catálogo[48].
- `load-combining-both`: si el atributo `namespace` o `schemaLocation` tiene una asignación de catálogo[48], entonces se usa la asignación. Si ambos atributos tienen asignaciones de catálogo[48], entonces es el valor de la opción `--schema-mapping` (opción XML/XSD[240]) decide qué asignación se utiliza. Si no hay ninguna asignación de catálogo[48], entonces se usa el atributo `schemaLocation`.
- `license-namespace-only`: se importa el espacio de nombres. No se importa el documento de esquema.

▼ schemalocation-hints

`--schemalocation-hints =` `load-by-schemalocation | load-by-namespace | load-combining-both | ignore`
Determina el comportamiento predeterminado de los atributos `xsi:schemaLocation` y `xsi:noNamespaceSchemaLocation`. Indica si se debe cargar un documento de esquema y, si así es, indica qué información debe utilizarse para encontrarlo. Valor predeterminado: `load-by-schemalocation`.

- **Valor predeterminado**: `load-by-schemalocation`. Este valor toma la URL de la ubicación del

 © 2017-2023 Altova GmbH

esquema[403] de los atributos `xsi:schemaLocation` y `xsi:noNamespaceSchemaLocation` de los documentos de instancia XML.

- El valor `load-by-namespace` toma la parte de espacio de nombres[403] del atributo `xsi:schemaLocation` y una cadena vacía en el caso del atributo `xsi:noNamespaceSchemaLocation` y encuentra el esquema por medio de una asignación de catálogo[48].
- Si usa el valor `load-combining-both` y el espacio de nombres o la URL tienen una asignación de catálogo[48], se usa dicha asignación. Si ambos tienen asignaciones de catálogo[48], el valor de la opción `schema-mapping` (opción XML/XSD[240]) decide qué asignación se utiliza. Si ni el espacio de nombres ni la URL tiene una asignación de catálogo, se usa la URL.
- El valor `ignore` ignora los atributos `xsi:schemaLocation` y `xsi:noNamespaceSchemaLocation`.

▼ schema-mapping

`--schema-mapping =` `prefer-schemalocation | prefer-namespace`
Si se usa la ubicación y el espacio de nombres para buscar el documento de esquema, esta opción indica cuál de ellos debe ser la opción preferida durante la búsqueda en el catálogo.
Si la opción `--schemalocation-hints` o la opción `--schema-imports` tiene el valor `load-combining-both` y si las partes de espacio de nombres y URL pertinentes tienen asignaciones de catálogo[48], entonces el valor de la opción especifica cuál de las dos asignaciones se utiliza (la asignación del espacio de nombres o de la URL: el valor `prefer-schemalocation` se refiere a la asignación de la URL). Valor predeterminado: `prefer-schemalocation`.

▼ script

`--script =` *`ARCHIVO`*
Una vez finalizada la validación, ejecuta el script Python. Para indicar más de un script basta con agregar la opción varias veces.

▼ script-api-version

`--api, --script-api-version =` `1; 2; 2.1 a 2.4; 2.4.1; 2.5 to 2.8; 2.8.1 a 2.8.6`
Especifica la versión de la API de Python a utilizar para el script. El valor predeterminado es la última versión, actualmente es **`2.8.6`**. En lugar de valores numéricos enteros como `1` y `2`, también puede utilizar los valores correspondientes `1.0` y `2.0`. Asimismo, puede utilizar el número de tres dígitos `2.5.0` en vez de dos (`2.5`). Consulte también el apartado Versiones de la API de Python[372].

▼ script-output

`--script-output =` *`FILE`*
Escribe el resultado estandard del script en el archivo nombres como *`FILE`*.

▼ script-param

`--script-param =` *`CLAVE:VALOR`*
Parámetros definidos por el usuario a los que se puede acceder durante la ejecución de scripts Python.

▼ streaming

`--streaming =` `true|false`

Habilita la transmisión por secuencias. En el modo de transmisión por secuencias, el almacenamiento de datos en memoria se reduce al mínimo y el procesamiento es más rápido. El inconveniente es que puede que no esté disponible cierta información que podría necesitar más adelante, como el modelo de datos del documento XML, por ejemplo. Si quiere evitar esto, debería deshabilitar el modo de transmisión por secuencias (dándole el valor `false` a la opción `--streaming`). Cuando use la opción `--script` con el comando `valxml-withxsd`, aconsejamos deshabilitar la transmisión por secuencias. Recuerde que la opción `--streaming` se ignora, si el valor de `--parallel-assessment` es `true`.
Valor predeterminado: `true`.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- xinclude

  `--xinclude =` `true|false`
  Habilita la compatibilidad con inclusiones XML (XInclude). Si el valor es `false`, los elementos XInclude `include` se ignoran. Valor predeterminado: `false`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- xml-mode

  `--xml-mode =` `wf|id|valid`
  Especifica el modo de procesamiento XML que debe utilizarse para el documento de instancia XML: `wf`=comprobación de formato; `id`=comprobación de formato con ID/IDREF; `valid`=validación. Valor predeterminado: `wf`. Recuerde que el valor `valid` exige que cada documento de instancia que se cargue durante el procesamiento haga referencia a una DTD. Si no existe ninguna DTD, se generará un error.

- xml-mode-for-schemas

  `--xml-mode-for-schemas =` `wf|id|valid`
  Especifica el modo de procesamiento XML que debe utilizarse para el documento de instancia XML: `wf`=comprobación de formato; `id`=comprobación de formato con ID/IDREF; `valid`=validación. Valor predeterminado: `wf`. Recuerde que el valor `valid` exige que cada documento de esquema que se cargue durante el procesamiento haga referencia a una DTD. Si no existe ninguna DTD, se generará un error.

- xsd

  `--xsd =` *`ARCHIVO`*
  Especifica qué esquemas XML deben utilizarse para la validación de documentos XML. Si quiere especificar más de un esquema, añada la opción varias veces.

- xsd-version

  `--xsd-version =` `1.0|1.1|detect`

  Especifica qué versión de la especificación Schema Definition Language (XSD) del W3C se debe usar. Valor predeterminado: `1.0`.

  Esta opción también puede ser útil si quiere ver en qué aspectos no es compatible un esquema 1.0 con la especificación 1.1. El valor `detect` es una característica de Altova. Permite detectar la versión del esquema XML (`1.0` o `1.1`) leyendo el valor del atributo `vc:minVersion` del elemento

 © 2017-2023 Altova GmbH

<xs:schema> del documento. Si el valor del atributo @vc:minVersion es 1.1, se entiende que la versión del esquema es 1.1. Si el atributo tiene otro valor que no sea 1.1 (o si no está presente el atributo @vc:minVersion), se entiende que la versión del esquema es 1.0.

Catálogos y recursos globales

catalog

--catalog = *ARCHIVO*

Especifica la ruta de acceso absoluta a un archivo de catálogo que no está en el archivo de catálogo raíz instalado. El valor predeterminado es la ruta de acceso absoluta del archivo de catálogo raíz instalado (<carpeta-instalación>\Altova\RaptorXMLServer2024\etc\RootCatalog.xml). Consulte el apartado Catálogos XML [48] para obtener más información.

user-catalog

--user-catalog = *ARCHIVO*

Especifica la ruta de acceso absoluta a un catálogo XML que debe utilizarse junto con el catálogo raíz. Consulte el apartado Catálogos XML [48] para obtener más información.

enable-globalresources

--enable-globalresources = true|false

Habilita la función de recursos globales [55]. Valor predeterminado: false.

Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en true.

globalresourceconfig [gc]

--gc | --globalresourceconfig = *VALOR*

Especifica la configuración activa del recurso global [55] (y habilita los recursos globales) [55].

globalresourcefile [gr]

--gr | --globalresourcefile = *ARCHIVO*

Especifica el archivo de recursos globales [55] (y habilita los recursos globales) [55].

Opciones comunes

error-format

--error-format = text|shortxml|longxml

Especifica el formato de la salida de error. Los valores posibles son formatos de texto, XML y XML detallado (longxml). Valor predeterminado: text.

error-limit

--error-limit = *N | unlimited*

Especifica el límite de errores con un valor comprendido entre 1 y 9999 o unlimited (ilimitado). El valor predeterminado es 100. Cuando se alcanza el límite de error, se detiene la validación. Esta opción es muy práctica a la hora de limitar el uso del procesador durante la validación o transformación.

- info-limit

  --info-limit = *N | unlimited*
  Indica el límite del mensaje de información dentro del rango `1-65535` or `unlimited`. Si se alcanza el límite de información indicado, el procesamiento continúa pero ya no se informa de más mensajes. El valor predeterminado es `100`.

- help

  --help
  Muestra el texto de ayuda para el comando. Por ejemplo `valany --h`. (Otra opción es usar el comando `help` con un argumento. Por ejemplo: `help valany`).

- listfile

  --listfile = `true|false`
  Si el valor es `true`, el argumento *ArchivoEntrada* del comando se entiende como un archivo de texto que contiene un nombre de archivo por línea. Otra opción es enumerar los archivos en la ILC, separados por un espacio. No obstante, recuerde que las ILC tienen un límite de caracteres. Además, no olvide que la opción `--listfile` solamente afecta a los argumentos y no a las opciones.
  Valor predeterminado: `false`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- log-output

  --log-output = *ARCHIVO*
  Escribe el registro de salida en la URL de archivo indicada. Compruebe que la ILC tiene permiso de escritura en la ubicación de destino.

- network-timeout

  --network-timeout = *VALOR*
  Indica el tiempo de espera en milisegundos para operaciones remotas de entrada y salida. Valor predeterminado: `40000`.

- recurse

  --recurse = `true|false`
  Esta opción se utiliza para seleccionar ficheros dentro de subdirectorios (incluso en archivos ZIP). Si el valor es `true`, el argumento *ArchivoEntrada* del comando seleccionará el fichero seleccionado también en los subdirectorios. Por ejemplo: `"test.zip|zip\test.xml"` seleccionará los ficheros llamados `test.xml` en todos los subdirectorios de la carpeta ZIP. Si quiere puede usar los caracteres comodín `*` y `?`. Por ejemplo: `*.xml` seleccionaría todos los ficheros de la carpeta ZIP que tengan la extensión `.xml`. Valor predeterminado: `false`
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- verbose

  --verbose = `true|false`

 © 2017-2023 Altova GmbH

Si el valor es `true`, se genera información adicional durante la validación. Valor predeterminado es `false`.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- verbose-output

  `--verbose-output =` *`ARCHIVO`*
  Escribe el resultado detallado en el *`ARCHIVO`* indicado.

- version

  `--version`
  Muestra el número de versión de RaptorXML Server. Si se utiliza con un comando, escriba la opción `--version` antes del comando.

- warning-limit

  `--warning-limit =` *`N | unlimited`*
  Especifica el límite de advertencia en el rango `1-65535` o `unlimited` (ilimitado). El procesamiento continua si se alcanza el límite pero no se registrarán más advertencias. El valor por defecto es `100`.

5.1.3 valdtd (dtd)

El comando **`valdtd | dtd`** valida uno o varios documentos DTD con la especificación XML 1.0 o XML 1.1.

Windows	**`RaptorXML valdtd	dtd [opciones]`** ***`ArchivoEntrada`***
Linux	**`raptorxml valdtd	dtd [opciones]`** ***`ArchivoEntrada`***
Mac	**`raptorxml valdtd	dtd [opciones]`** ***`ArchivoEntrada`***

- El argumento *`ArchivoEntrada`* es el documento DTD que debe validarse.
- Si desea validar varios documentos, tiene dos opciones: (i) enumerar los archivos que se deben validar en la línea de comandos, separados por espacios o (ii) enumerar los archivos que se deben validar en un archivo de texto (`.txt`) donde aparece un nombre de archivo por línea y dar este archivo de texto como argumento *`ArchivoEntrada`* junto con la opción `--listfile` 238 con valor `true` (*ver lista de opciones más abajo*).

Ejemplos

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

- Mayúsculas/minúsculas y barras en la línea de comandos

RaptorXML (y **RaptorXMLServer** para comandos administrativos) *en Windows*
raptorxml (y **raptorxmlserver** para comandos administrativos) *en Windows y Unix (Linux, Mac)*

* Las minúsculas (raptorxml y raptorxmlserver) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que las mayúsculas (RaptorXML) funcionan solamente en Windows y Mac.
* Use la barra diagonal en Linux y Mac y la barra diagonal inversa en Windows.

▼ Barra diagonal inversa y espacios en sistemas Windows

En sistemas Windows: si hay espacios o caracteres especiales (por ejemplo en los nombres de archivos o de carpetas, empresas, personas o productos) debe usar comillas: por ejemplo, **"Mi archivo"**. Sin embargo, debe tener en cuenta que una barra diagonal inversa seguida de comillas dobles (por ejemplo, **"C:\Mi Directorio\"**) es posible que no se lea correctamente. Esto se debe a que la barra diagonal inversa también se usa para empezar secuencias de escape y la secuencia de escape **\"** también usa las comillas dobles. Para evitar secuencia de caracteres puede añadir otra barra diagonal inversa: **\\"**. En resumen: si necesita escribir una ruta que contenga espacios y una barra diagonal inversa, esta es la mejor forma de hacerlo: **"C:\Mi Directorio\\"**.

Opciones

Las opciones del comando aparecen más abajo divididas en grupos. Los valores se pueden dar sin comillas excepto en estos dos casos: (i) cuando la cadena de valor contiene espacios y (ii) cuando en la descripción de la opción se indique explícitamente que es necesario el uso de comillas. Si una opción toma un valor booleano y no se indica ningún valor, entonces el valor predeterminado de la opción es TRUE. Use la opción **--h, --help** para ver información sobre el comando.

▼ Validación y procesamiento

▼ listfile

--listfile = true|false
Si el valor es true, el argumento *ArchivoEntrada* del comando se entiende como un archivo de texto que contiene un nombre de archivo por línea. Otra opción es enumerar los archivos en la ILC, separados por un espacio. No obstante, recuerde que las ILC tienen un límite de caracteres. Además, no olvide que la opción --listfile solamente afecta a los argumentos y no a las opciones.
Valor predeterminado: false.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en true.

▼ recurse

--recurse = true|false
Esta opción se utiliza para seleccionar ficheros dentro de subdirectorios (incluso en archivos ZIP). Si el valor es true, el argumento *ArchivoEntrada* del comando seleccionará el fichero seleccionado también en los subdirectorios. Por ejemplo: "test.zip|zip\test.xml" seleccionará los ficheros llamados test.xml en todos los subdirectorios de la carpeta ZIP. Si quiere puede usar los caracteres comodín * y ?. Por ejemplo: *.xml seleccionaría todos los ficheros de la carpeta ZIP que tengan la extensión .xml. Valor predeterminado: false
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en true.

 © 2017-2023 Altova GmbH

- script

  --script = *ARCHIVO*
  Una vez finalizada la validación, ejecuta el script Python. Para indicar más de un script basta con agregar la opción varias veces.

- script-api-version

  --api, --script-api-version = `1; 2; 2.1 a 2.4; 2.4.1; 2.5 to 2.8; 2.8.1 a 2.8.6`
  Especifica la versión de la API de Python a utilizar para el script. El valor predeterminado es la última versión, actualmente es **`2.8.6`**. En lugar de valores numéricos enteros como `1` y `2`, también puede utilizar los valores correspondientes `1.0` y `2.0`. Asimismo, puede utilizar el número de tres dígitos `2.5.0` en vez de dos (`2.5`). Consulte también el apartado Versiones de la API de Python 372.

- script-output

  --script-output = *FILE*
  Escribe el resultado estandard del script en el archivo nombres como *FILE*.

- script-param

  --script-param = *CLAVE:VALOR*
  Parámetros definidos por el usuario a los que se puede acceder durante la ejecución de scripts Python.

- Catálogos y recursos globales
  - catalog

    --catalog = *ARCHIVO*
    Especifica la ruta de acceso absoluta a un archivo de catálogo que no está en el archivo de catálogo raíz instalado. El valor predeterminado es la ruta de acceso absoluta del archivo de catálogo raíz instalado (`<carpeta-instalación>\Altova\RaptorXMLServer2024\etc\RootCatalog.xml`). Consulte el apartado Catálogos XML 48 para obtener más información.

  - user-catalog

    --user-catalog = *ARCHIVO*
    Especifica la ruta de acceso absoluta a un catálogo XML que debe utilizarse junto con el catálogo raíz. Consulte el apartado Catálogos XML 48 para obtener más información.

  - enable-globalresources

    --enable-globalresources = `true|false`
    Habilita la función de recursos globales 55. Valor predeterminado: `false`.
    Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

  - globalresourceconfig [gc]

    --gc | --globalresourceconfig = *VALOR*
    Especifica la configuración activa del recurso global 55 (y habilita los recursos globales) 55.

- globalresourcefile [gr]

  `--gr | --globalresourcefile =` *`ARCHIVO`*
  Especifica el archivo de recursos globales[55] (y habilita los recursos globales)[55].

- Opciones comunes
  - error-format

    `--error-format =` `text|shortxml|longxml`
    Especifica el formato de la salida de error. Los valores posibles son formatos de texto, XML y XML detallado (`longxml`). Valor predeterminado: `text`.

  - error-limit

    `--error-limit =` *`N | unlimited`*
    Especifica el límite de errores con un valor comprendido entre `1` y `9999` o `unlimited` (ilimitado). El valor predeterminado es `100`. Cuando se alcanza el límite de error, se detiene la validación. Esta opción es muy práctica a la hora de limitar el uso del procesador durante la validación o transformación.

  - info-limit

    `--info-limit =` *`N | unlimited`*
    Indica el límite del mensaje de información dentro del rango `1-65535` or `unlimited`. Si se alcanza el límite de información indicado, el procesamiento continúa pero ya no se informa de más mensajes. El valor predeterminado es `100`.

  - help

    `--help`
    Muestra el texto de ayuda para el comando. Por ejemplo `valany --h`. (Otra opción es usar el comando `help` con un argumento. Por ejemplo: `help valany`).

  - listfile

    `--listfile =` `true|false`
    Si el valor es `true`, el argumento *`ArchivoEntrada`* del comando se entiende como un archivo de texto que contiene un nombre de archivo por línea. Otra opción es enumerar los archivos en la ILC, separados por un espacio. No obstante, recuerde que las ILC tienen un límite de caracteres. Además, no olvide que la opción `--listfile` solamente afecta a los argumentos y no a las opciones.
    Valor predeterminado: `false`.
    Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

  - log-output

    `--log-output =` *`ARCHIVO`*
    Escribe el registro de salida en la URL de archivo indicada. Compruebe que la ILC tiene permiso de escritura en la ubicación de destino.

© 2017-2023 Altova GmbH

- network-timeout

  --network-timeout = *VALOR*
  Indica el tiempo de espera en milisegundos para operaciones remotas de entrada y salida. Valor predeterminado: `40000`.

- recurse

  --recurse = `true|false`
  Esta opción se utiliza para seleccionar ficheros dentro de subdirectorios (incluso en archivos ZIP). Si el valor es `true`, el argumento *ArchivoEntrada* del comando seleccionará el fichero seleccionado también en los subdirectorios. Por ejemplo: `"test.zip|zip\test.xml"` seleccionará los ficheros llamados `test.xml` en todos los subdirectorios de la carpeta ZIP. Si quiere puede usar los caracteres comodín `*` y `?`. Por ejemplo: `*.xml` seleccionaría todos los ficheros de la carpeta ZIP que tengan la extensión `.xml`. Valor predeterminado: `false`
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- verbose

  --verbose = `true|false`
  Si el valor es `true`, se genera información adicional durante la validación. Valor predeterminado es `false`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- verbose-output

  --verbose-output = *ARCHIVO*
  Escribe el resultado detallado en el *ARCHIVO* indicado.

- version

  --version
  Muestra el número de versión de RaptorXML Server. Si se utiliza con un comando, escriba la opción `--version` antes del comando.

- warning-limit

  --warning-limit = *N | unlimited*
  Especifica el límite de advertencia en el rango `1-65535` o `unlimited` (ilimitado). El procesamiento continua si se alcanza el límite pero no se registrarán más advertencias. El valor por defecto es `100`.

5.1.4 valxsd (xsd)

El comando **valxsd | xsd** valida uno o varios documentos de esquema XML (documentos XSD) con la especificación XML Schema Definition Language (XSD) 1.0 o 1.1 del W3C. Tenga en cuenta que lo que se valida con la especificación XML Schema es un esquema XML y no un documento XML de instancia con un esquema XML.

Windows	**RaptorXML valxsd | xsd [opciones]** ***ArchivoEntrada***
Linux	**raptorxml valxsd | xsd [opciones]** ***ArchivoEntrada***
Mac	**raptorxml valxsd | xsd [opciones]** ***ArchivoEntrada***

- El argumento *ArchivoEntrada* suministra el documento de esquema XML que debe validarse. La opción `--xsd-version=1.0|1.1|detect` 240 indica la versión XSD con la que debe validarse, siendo `1.0` su valor predeterminado.
- Si desea validar varios documentos, tiene dos opciones: (i) enumerar los archivos que se deben validar en la línea de comandos, separados por espacios o (ii) enumerar los archivos que se deben validar en un archivo de texto (`.txt`) donde aparece un nombre de archivo por línea y dar este archivo de texto como argumento *ArchivoEntrada* junto con la opción `--listfile` 238 con valor `true` (*ver lista de opciones más abajo*).

Ejemplos

Ejemplos del comando **valxsd**:

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

▼ Mayúsculas/minúsculas y barras en la línea de comandos

RaptorXML (y **RaptorXMLServer** para comandos administrativos) *en Windows*
raptorxml (y **raptorxmlserver** para comandos administrativos) *en Windows y Unix (Linux, Mac)*

* Las minúsculas (`raptorxml` y `raptorxmlserver`) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que las mayúsculas (`RaptorXML`) funcionan solamente en Windows y Mac.
* Use la barra diagonal en Linux y Mac y la barra diagonal inversa en Windows.

▼ Barra diagonal inversa y espacios en sistemas Windows

En sistemas Windows: si hay espacios o caracteres especiales (por ejemplo en los nombres de archivos o de carpetas, empresas, personas o productos) debe usar comillas: por ejemplo, **"Mi archivo"**. Sin embargo, debe tener en cuenta que una barra diagonal inversa seguida de comillas dobles (por ejemplo, **"C:\Mi Directorio\"**) es posible que no se lea correctamente. Esto se debe a que la barra diagonal inversa también se usa para empezar secuencias de escape y la secuencia de escape **\"** también usa las comillas dobles. Para evitar secuencia de caracteres puede añadir otra barra diagonal inversa: **\\"**. En resumen: si necesita escribir una ruta que contenga espacios y una barra diagonal inversa, esta es la mejor forma de hacerlo: **"C:\Mi Directorio\\"**.

Opciones

Las opciones del comando aparecen más abajo divididas en grupos. Los valores se pueden dar sin comillas excepto en estos dos casos: (i) cuando la cadena de valor contiene espacios y (ii) cuando en la descripción de la opción se indique explícitamente que es necesario el uso de comillas. Si una opción toma un valor booleano y no se indica ningún valor, entonces el valor predeterminado de la opción es `TRUE`. Use la opción **--h, --help** para ver información sobre el comando.

© 2017-2023 Altova GmbH

- Validación y procesamiento
  - ct-restrict-mode

    --ct-restrict-mode = 1.0|1.1|default
    Especifica cómo comprobar restricciones de tipo complejo. Un valor de **1.0** comprueba restricciones de tipo complejo conforme a lo definido en la especificación XSD 1.0 (incluso estando en modo de validación XSD 1.1). Un valor de **1.1** comprueba restricciones de tipo complejo conforme a los definido en la especificación XSD 1.1 (incluso estando en modo de validación XSD 1.0). Un valor de **default** comprueba restricciones de tipo complejo conforme a lo definido en la especificación XSD del modo de validación habilitado en ese momento (1.0 o 1.1). El valor predeterminado es **default**.
  - listfile

    --listfile = true|false
    Si el valor es true, el argumento *ArchivoEntrada* del comando se entiende como un archivo de texto que contiene un nombre de archivo por línea. Otra opción es enumerar los archivos en la ILC, separados por un espacio. No obstante, recuerde que las ILC tienen un límite de caracteres. Además, no olvide que la opción --listfile solamente afecta a los argumentos y no a las opciones.
    Valor predeterminado: false.
    Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en true.
  - recurse

    --recurse = true|false
    Esta opción se utiliza para seleccionar ficheros dentro de subdirectorios (incluso en archivos ZIP). Si el valor es true, el argumento *ArchivoEntrada* del comando seleccionará el fichero seleccionado también en los subdirectorios. Por ejemplo: "test.zip|zip\test.xml" seleccionará los ficheros llamados test.xml en todos los subdirectorios de la carpeta ZIP. Si quiere puede usar los caracteres comodín * y ?. Por ejemplo: *.xml seleccionaría todos los ficheros de la carpeta ZIP que tengan la extensión .xml. Valor predeterminado: false
    Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en true.
  - report-import-namespace-mismatch-as-warning

    --report-import-namespace-mismatch-as-warning = true|false
    Informa de los errores de disparidad del espacio de nombres o del espacio de nombres de destino al importar esquemas con **xs:import** como advertencias en vez de como errores. El valor predeterminado es false.
    Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en true.
  - schema-imports

    --schema-imports = load-by-schemalocation | load-preferring-schemalocation | load-by-namespace | load-combining-both | license-namespace-only
    Esta opción indica el comportamiento de los elementos xs:import. Cada uno de estos elementos tiene un atributo opcional namespace y un atributo opcional schemaLocation: <import namespace="unEspacioNombres" schemaLocation="unaURL">. La opción indica si se debe cargar un documento de esquema o solo autorizar a un espacio de nombres. Si la opción indica que se debe cargar un documento de esquema, entonces indica también qué información debe utilizarse

para encontrar el documento de esquema. Valor predeterminado: `load-preferring-schemalocation`.

- `load-by-schemalocation`: el valor del atributo `schemaLocation` se utiliza para buscar el esquema, teniendo en cuenta las asignaciones de catálogo[48]. Si está presente el atributo `namespace`, se importa el espacio de nombres (con licencia).
- `load-preferring-schemalocation`: si está presente, se utiliza el atributo `schemaLocation` teniendo en cuenta las asignaciones de catálogo[48]. Si no está presente el atributo `schemaLocation`, entonces se usa el valor del atributo `namespace` a través de las asignaciones de catálogo[48]. Este es el **valor predeterminado**.
- `load-by-namespace`: el valor del atributo `namespace` se utiliza para buscar el esquema por medio de una asignación de catálogo[48].
- `load-combining-both`: si el atributo `namespace` o `schemaLocation` tiene una asignación de catálogo[48], entonces se usa la asignación. Si ambos atributos tienen asignaciones de catálogo[48], entonces es el valor de la opción `--schema-mapping` (opción XML/XSD[240]) decide qué asignación se utiliza. Si no hay ninguna asignación de catálogo[48], entonces se usa el atributo `schemaLocation`.
- `license-namespace-only`: se importa el espacio de nombres. No se importa el documento de esquema.

▼ schemalocation-hints

`--schemalocation-hints =` `load-by-schemalocation | load-by-namespace | load-combining-both | ignore`
Determina el comportamiento predeterminado de los atributos `xsi:schemaLocation` y `xsi:noNamespaceSchemaLocation`. Indica si se debe cargar un documento de esquema y, si así es, indica qué información debe utilizarse para encontrarlo. Valor predeterminado: `load-by-schemalocation`.

- **Valor predeterminado**: `load-by-schemalocation`. Este valor toma la URL de la ubicación del esquema[403] de los atributos `xsi:schemaLocation` y `xsi:noNamespaceSchemaLocation` de los documentos de instancia XML.

- El valor `load-by-namespace` toma la parte de espacio de nombres[403] del atributo `xsi:schemaLocation` y una cadena vacía en el caso del atributo `xsi:noNamespaceSchemaLocation` y encuentra el esquema por medio de una asignación de catálogo[48].

- Si usa el valor `load-combining-both` y el espacio de nombres o la URL tienen una asignación de catálogo[48], se usa dicha asignación. Si ambos tienen asignaciones de catálogo[48], el valor de la opción `schema-mapping` (opción XML/XSD[240]) decide qué asignación se utiliza. Si ni el espacio de nombres ni la URL tiene una asignación de catálogo, se usa la URL.

- El valor `ignore` ignora los atributos `xsi:schemaLocation` y `xsi:noNamespaceSchemaLocation`.

▼ schema-mapping

`--schema-mapping =` `prefer-schemalocation | prefer-namespace`
Si se usa la ubicación y el espacio de nombres para buscar el documento de esquema, esta opción indica cuál de ellos debe ser la opción preferida durante la búsqueda en el catálogo.
Si la opción `--schemalocation-hints` o la opción `--schema-imports` tiene el valor `load-combining-both` y si las partes de espacio de nombres y URL pertinentes tienen asignaciones de catálogo[48], entonces el valor de la opción especifica cuál de las dos asignaciones se utiliza (la

 © 2017-2023 Altova GmbH

asignación del espacio de nombres o de la URL: el valor `prefer-schemalocation` se refiere a la asignación de la URL). Valor predeterminado: `prefer-schemalocation`.

- script

  `--script =` *`ARCHIVO`*
  Una vez finalizada la validación, ejecuta el script Python. Para indicar más de un script basta con agregar la opción varias veces.

- script-api-version

  `--api, --script-api-version =` `1; 2; 2.1 a 2.4; 2.4.1; 2.5 to 2.8; 2.8.1 a 2.8.6`
  Especifica la versión de la API de Python a utilizar para el script. El valor predeterminado es la última versión, actualmente es **`2.8.6`**. En lugar de valores numéricos enteros como `1` y `2`, también puede utilizar los valores correspondientes `1.0` y `2.0`. Asimismo, puede utilizar el número de tres dígitos `2.5.0` en vez de dos (`2.5`). Consulte también el apartado Versiones de la API de Python 372.

- script-output

  `--script-output =` *`FILE`*
  Escribe el resultado estandard del script en el archivo nombres como *`FILE`*.

- script-param

  `--script-param =` *`CLAVE:VALOR`*
  Parámetros definidos por el usuario a los que se puede acceder durante la ejecución de scripts Python.

- xinclude

  `--xinclude =` `true|false`
  Habilita la compatibilidad con inclusiones XML (XInclude). Si el valor es `false`, los elementos XInclude `include` se ignoran. Valor predeterminado: `false`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- xml-mode-for-schemas

  `--xml-mode-for-schemas =` `wf|id|valid`
  Especifica el modo de procesamiento XML que debe utilizarse para el documento de instancia XML: `wf`=comprobación de formato; `id`=comprobación de formato con ID/IDREF; `valid`=validación. Valor predeterminado: `wf`. Recuerde que el valor `valid` exige que cada documento de esquema que se cargue durante el procesamiento haga referencia a una DTD. Si no existe ninguna DTD, se generará un error.

- xsd-version

  `--xsd-version =` `1.0|1.1|detect`

  Especifica qué versión de la especificación Schema Definition Language (XSD) del W3C se debe usar. Valor predeterminado: `1.0`.

  Esta opción también puede ser útil si quiere ver en qué aspectos no es compatible un esquema 1.0 con la especificación 1.1. El valor `detect` es una característica de Altova. Permite detectar la versión

del esquema XML (`1.0` o `1.1`) leyendo el valor del atributo `vc:minVersion` del elemento `<xs:schema>` del documento. Si el valor del atributo `@vc:minVersion` es `1.1`, se entiende que la versión del esquema es `1.1`. Si el atributo tiene otro valor que no sea `1.1` (o si no está presente el atributo `@vc:minVersion`), se entiende que la versión del esquema es `1.0`.

Catálogos y recursos globales

catalog

`--catalog =` *ARCHIVO*
Especifica la ruta de acceso absoluta a un archivo de catálogo que no está en el archivo de catálogo raíz instalado. El valor predeterminado es la ruta de acceso absoluta del archivo de catálogo raíz instalado (`<carpeta-instalación>\Altova\RaptorXMLServer2024\etc\RootCatalog.xml`). Consulte el apartado Catálogos XML [48] para obtener más información.

user-catalog

`--user-catalog =` *ARCHIVO*
Especifica la ruta de acceso absoluta a un catálogo XML que debe utilizarse junto con el catálogo raíz. Consulte el apartado Catálogos XML [48] para obtener más información.

enable-globalresources

`--enable-globalresources =` `true|false`
Habilita la función de recursos globales [55]. Valor predeterminado: `false`.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

globalresourceconfig [gc]

`--gc | --globalresourceconfig =` *VALOR*
Especifica la configuración activa del recurso global [55] (y habilita los recursos globales) [55].

globalresourcefile [gr]

`--gr | --globalresourcefile =` *ARCHIVO*
Especifica el archivo de recursos globales [55] (y habilita los recursos globales) [55].

Opciones comunes

error-format

`--error-format =` `text|shortxml|longxml`
Especifica el formato de la salida de error. Los valores posibles son formatos de texto, XML y XML detallado (`longxml`). Valor predeterminado: `text`.

error-limit

`--error-limit =` *N | unlimited*
Especifica el límite de errores con un valor comprendido entre `1` y `9999` o `unlimited` (ilimitado). El valor predeterminado es `100`. Cuando se alcanza el límite de error, se detiene la validación. Esta opción es muy práctica a la hora de limitar el uso del procesador durante la validación o

© 2017-2023 Altova GmbH

transformación.

- info-limit

  --info-limit = *N | unlimited*
  Indica el límite del mensaje de información dentro del rango `1-65535` or `unlimited`. Si se alcanza el límite de información indicado, el procesamiento continúa pero ya no se informa de más mensajes. El valor predeterminado es `100`.

- help

  --help
  Muestra el texto de ayuda para el comando. Por ejemplo `valany --h`. (Otra opción es usar el comando `help` con un argumento. Por ejemplo: `help valany`).

- listfile

  --listfile = `true|false`
  Si el valor es `true`, el argumento *ArchivoEntrada* del comando se entiende como un archivo de texto que contiene un nombre de archivo por línea. Otra opción es enumerar los archivos en la ILC, separados por un espacio. No obstante, recuerde que las ILC tienen un límite de caracteres. Además, no olvide que la opción `--listfile` solamente afecta a los argumentos y no a las opciones.
  Valor predeterminado: `false`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- log-output

  --log-output = *ARCHIVO*
  Escribe el registro de salida en la URL de archivo indicada. Compruebe que la ILC tiene permiso de escritura en la ubicación de destino.

- network-timeout

  --network-timeout = *VALOR*
  Indica el tiempo de espera en milisegundos para operaciones remotas de entrada y salida. Valor predeterminado: `40000`.

- recurse

  --recurse = `true|false`
  Esta opción se utiliza para seleccionar ficheros dentro de subdirectorios (incluso en archivos ZIP). Si el valor es `true`, el argumento *ArchivoEntrada* del comando seleccionará el fichero seleccionado también en los subdirectorios. Por ejemplo: `"test.zip|zip\test.xml"` seleccionará los ficheros llamados `test.xml` en todos los subdirectorios de la carpeta ZIP. Si quiere puede usar los caracteres comodín `*` y `?`. Por ejemplo: `*.xml` seleccionaría todos los ficheros de la carpeta ZIP que tengan la extensión `.xml`. Valor predeterminado: `false`
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- verbose

  --verbose = `true|false`

Si el valor es `true`, se genera información adicional durante la validación. Valor predeterminado es `false`.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- verbose-output

  `--verbose-output =` *`ARCHIVO`*
  Escribe el resultado detallado en el *`ARCHIVO`* indicado.

- version

  `--version`
  Muestra el número de versión de RaptorXML Server. Si se utiliza con un comando, escriba la opción `--version` antes del comando.

- warning-limit

  `--warning-limit =` *`N | unlimited`*
  Especifica el límite de advertencia en el rango `1-65535` o `unlimited` (ilimitado). El procesamiento continua si se alcanza el límite pero no se registrarán más advertencias. El valor por defecto es `100`.

© 2017-2023 Altova GmbH

5.2 Comandos para comprobar el formato

Los comandos de comprobación de formato sirven para comprobar si el formato de documentos XML y DTD es correcto.

wfxml 83	Comprueba si el documento XML tiene un formato correcto.
wfdtd 87	Comprueba si el documento DTD tiene un formato correcto.
wfany 209	Comprueba si el documento XML o DTD tiene un formato correcto. El tipo se detecta automáticamente.

5.2.1 wfxml

El comando **wfxml** revisa uno o varios documentos XML y comprueba si su formato es correcto según la especificación XML 1.0 o XML 1.1.

Windows **RaptorXML wfxml [opciones] *ArchivoEntrada***

Linux **raptorxml wfxml [opciones] *ArchivoEntrada***

Mac **raptorxml wfxml [opciones] *ArchivoEntrada***

El argumento *ArchivoEntrada* es el documento XML cuyo formato debe comprobarse.

Si desea comprobar el formato de varios documentos, tiene dos opciones: (i) enumerar los archivos que se deben revisar en la línea de comandos, separados por espacios o (ii) enumerar los archivos que se deben revisar en un archivo de texto (`.txt`) donde aparece un nombre de archivo por línea y dar este archivo de texto como argumento *ArchivoEntrada* junto con la opción `--listfile` 238 con valor `true` (*ver lista de opciones más abajo*).

Ejemplos

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

▼ Mayúsculas/minúsculas y barras en la línea de comandos

RaptorXML (y **RaptorXMLServer** para comandos administrativos) *en Windows*
raptorxml (y **raptorxmlserver** para comandos administrativos) *en Windows y Unix (Linux, Mac)*

* Las minúsculas (`raptorxml` y `raptorxmlserver`) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que las mayúsculas (`RaptorXML`) funcionan solamente en Windows y Mac.
* Use la barra diagonal en Linux y Mac y la barra diagonal inversa en Windows.

▼ Barra diagonal inversa y espacios en sistemas Windows

En sistemas Windows: si hay espacios o caracteres especiales (por ejemplo en los nombres de archivos o de carpetas, empresas, personas o productos) debe usar comillas: por ejemplo, **"Mi archivo"**. Sin embargo, debe tener en cuenta que una barra diagonal inversa seguida de comillas dobles (por ejemplo, **"C:\Mi Directorio\"**) es posible que no se lea correctamente. Esto se debe a que la barra diagonal inversa también se usa para empezar secuencias de escape y la secuencia de escape **\"** también usa las comillas dobles. Para evitar secuencia de caracteres puede añadir otra barra diagonal inversa: **\\"**. En resumen: si necesita escribir una ruta que contenga espacios y una barra diagonal inversa, esta es la mejor forma de hacerlo: **"C:\Mi Directorio\\"**.

Opciones

Las opciones del comando aparecen más abajo divididas en grupos. Los valores se pueden dar sin comillas excepto en estos dos casos: (i) cuando la cadena de valor contiene espacios y (ii) cuando en la descripción de la opción se indique explícitamente que es necesario el uso de comillas. Si una opción toma un valor booleano y no se indica ningún valor, entonces el valor predeterminado de la opción es `TRUE`. Use la opción **--h, --help** para ver información sobre el comando.

- Validación y procesamiento
  - dtd

    --dtd = *ARCHIVO*
    Especifica el documento DTD externo que debe utilizarse para la validación. Si en el documento XML hay una referencia a una DTD externa, esta opción de la ILC reemplaza a la referencia externa.

  - listfile

    --listfile = `true|false`
    Si el valor es `true`, el argumento *ArchivoEntrada* del comando se entiende como un archivo de texto que contiene un nombre de archivo por línea. Otra opción es enumerar los archivos en la ILC, separados por un espacio. No obstante, recuerde que las ILC tienen un límite de caracteres. Además, no olvide que la opción `--listfile` solamente afecta a los argumentos y no a las opciones.
    Valor predeterminado: `false`.
    Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

  - namespaces

    --namespaces = `true|false`
    Habilita el procesamiento preparado para espacios de nombres. Esta opción es muy útil si quiere buscar en la instancia XML errores resultantes de espacios de nombres erróneos.
    Valor predeterminado: `false`.
    Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

  - recurse

    --recurse = `true|false`
    Esta opción se utiliza para seleccionar ficheros dentro de subdirectorios (incluso en archivos ZIP). Si el valor es `true`, el argumento *ArchivoEntrada* del comando seleccionará el fichero seleccionado también en los subdirectorios. Por ejemplo: `"test.zip|zip\test.xml"` seleccionará los ficheros

 © 2017-2023 Altova GmbH

llamados `test.xml` en todos los subdirectorios de la carpeta ZIP. Si quiere puede usar los caracteres comodín `*` y `?`. Por ejemplo: `*.xml` seleccionaría todos los ficheros de la carpeta ZIP que tengan la extensión `.xml`. Valor predeterminado: `false`
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- streaming

  `--streaming =` `true|false`
  Habilita la transmisión por secuencias. En el modo de transmisión por secuencias, el almacenamiento de datos en memoria se reduce al mínimo y el procesamiento es más rápido. El inconveniente es que puede que no esté disponible cierta información que podría necesitar más adelante, como el modelo de datos del documento XML, por ejemplo. Si quiere evitar esto, debería deshabilitar el modo de transmisión por secuencias (dándole el valor `false` a la opción `--streaming`). Cuando use la opción `--script` con el comando `valxml-withxsd`, aconsejamos deshabilitar la transmisión por secuencias. Recuerde que la opción `--streaming` se ignora, si el valor de `--parallel-assessment` es `true`.
  Valor predeterminado: `true`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- Catálogos y recursos globales
  - catalog

    `--catalog =` *ARCHIVO*
    Especifica la ruta de acceso absoluta a un archivo de catálogo que no está en el archivo de catálogo raíz instalado. El valor predeterminado es la ruta de acceso absoluta del archivo de catálogo raíz instalado (`<carpeta-instalación>\Altova\RaptorXMLServer2024\etc\RootCatalog.xml`). Consulte el apartado Catálogos XML [48] para obtener más información.

  - user-catalog

    `--user-catalog =` *ARCHIVO*
    Especifica la ruta de acceso absoluta a un catálogo XML que debe utilizarse junto con el catálogo raíz. Consulte el apartado Catálogos XML [48] para obtener más información.

  - enable-globalresources

    `--enable-globalresources =` `true|false`
    Habilita la función de recursos globales [55]. Valor predeterminado: `false`.
    Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

  - globalresourceconfig [gc]

    `--gc | --globalresourceconfig =` *VALOR*
    Especifica la configuración activa del recurso global [55] (y habilita los recursos globales) [55].

  - globalresourcefile [gr]

    `--gr | --globalresourcefile =` *ARCHIVO*

Especifica el archivo de recursos globales[55] (y habilita los recursos globales)[55].

- Opciones comunes
  - error-format

    --error-format = `text|shortxml|longxml`
    Especifica el formato de la salida de error. Los valores posibles son formatos de texto, XML y XML detallado (`longxml`). Valor predeterminado: `text`.

  - error-limit

    --error-limit = *N | unlimited*
    Especifica el límite de errores con un valor comprendido entre `1` y `9999` o `unlimited` (ilimitado). El valor predeterminado es `100`. Cuando se alcanza el límite de error, se detiene la validación. Esta opción es muy práctica a la hora de limitar el uso del procesador durante la validación o transformación.

  - info-limit

    --info-limit = *N | unlimited*
    Indica el límite del mensaje de información dentro del rango `1-65535` or `unlimited`. Si se alcanza el límite de información indicado, el procesamiento continúa pero ya no se informa de más mensajes. El valor predeterminado es `100`.

  - help

    --help
    Muestra el texto de ayuda para el comando. Por ejemplo `valany --h`. (Otra opción es usar el comando `help` con un argumento. Por ejemplo: `help valany`).

  - listfile

    --listfile = `true|false`
    Si el valor es `true`, el argumento *ArchivoEntrada* del comando se entiende como un archivo de texto que contiene un nombre de archivo por línea. Otra opción es enumerar los archivos en la ILC, separados por un espacio. No obstante, recuerde que las ILC tienen un límite de caracteres. Además, no olvide que la opción `--listfile` solamente afecta a los argumentos y no a las opciones.
    Valor predeterminado: `false`.
    Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

  - log-output

    --log-output = *ARCHIVO*
    Escribe el registro de salida en la URL de archivo indicada. Compruebe que la ILC tiene permiso de escritura en la ubicación de destino.

  - network-timeout

    --network-timeout = *VALOR*
    Indica el tiempo de espera en milisegundos para operaciones remotas de entrada y salida. Valor

 © 2017-2023 Altova GmbH

predeterminado: `40000`.

- recurse

  `--recurse =` `true|false`
  Esta opción se utiliza para seleccionar ficheros dentro de subdirectorios (incluso en archivos ZIP). Si el valor es `true`, el argumento *`ArchivoEntrada`* del comando seleccionará el fichero seleccionado también en los subdirectorios. Por ejemplo: `"test.zip|zip\test.xml"` seleccionará los ficheros llamados `test.xml` en todos los subdirectorios de la carpeta ZIP. Si quiere puede usar los caracteres comodín `*` y `?`. Por ejemplo: `*.xml` seleccionaría todos los ficheros de la carpeta ZIP que tengan la extensión `.xml`. Valor predeterminado: `false`
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- verbose

  `--verbose =` `true|false`
  Si el valor es `true`, se genera información adicional durante la validación. Valor predeterminado es `false`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- verbose-output

  `--verbose-output =` *`ARCHIVO`*
  Escribe el resultado detallado en el *`ARCHIVO`* indicado.

- version

  `--version`
  Muestra el número de versión de RaptorXML Server. Si se utiliza con un comando, escriba la opción `--version` antes del comando.

- warning-limit

  `--warning-limit =` *`N | unlimited`*
  Especifica el límite de advertencia en el rango `1-65535` o `unlimited` (ilimitado). El procesamiento continua si se alcanza el límite pero no se registrarán más advertencias. El valor por defecto es `100`.

5.2.2 wfdtd

El comando **`wfdtd`** revisa uno o varios documentos DTD y comprueba si su formato es correcto según la especificación XML 1.0 o XML 1.1.

Windows	**`RaptorXML wfdtd [opciones] ArchivoEntrada`**
Linux	**`raptorxml wfdtd [opciones] ArchivoEntrada`**

Mac **raptorxml wfdtd [opciones] *ArchivoEntrada***

El argumento *ArchivoEntrada* es el documento DTD cuyo formato debe comprobarse.

Si desea comprobar el formato de varios documentos, tiene dos opciones: (i) enumerar los archivos que se deben revisar en la línea de comandos, separados por espacios o (ii) enumerar los archivos que se deben revisar en un archivo de texto (`.txt`) donde aparece un nombre de archivo por línea y dar este archivo de texto como argumento *ArchivoEntrada* junto con la opción `--listfile` 238 con valor `true` (*ver lista de opciones más abajo*).

Ejemplos

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

▼ Mayúsculas/minúsculas y barras en la línea de comandos

RaptorXML (y **RaptorXMLServer** para comandos administrativos) *en Windows*
raptorxml (y **raptorxmlserver** para comandos administrativos) *en Windows y Unix (Linux, Mac)*

* Las minúsculas (`raptorxml` y `raptorxmlserver`) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que las mayúsculas (`RaptorXML`) funcionan solamente en Windows y Mac.
* Use la barra diagonal en Linux y Mac y la barra diagonal inversa en Windows.

▼ Barra diagonal inversa y espacios en sistemas Windows

En sistemas Windows: si hay espacios o caracteres especiales (por ejemplo en los nombres de archivos o de carpetas, empresas, personas o productos) debe usar comillas: por ejemplo, **"Mi archivo"**. Sin embargo, debe tener en cuenta que una barra diagonal inversa seguida de comillas dobles (por ejemplo, **"C:\Mi Directorio\"**) es posible que no se lea correctamente. Esto se debe a que la barra diagonal inversa también se usa para empezar secuencias de escape y la secuencia de escape **\"** también usa las comillas dobles. Para evitar secuencia de caracteres puede añadir otra barra diagonal inversa: **\\"**. En resumen: si necesita escribir una ruta que contenga espacios y una barra diagonal inversa, esta es la mejor forma de hacerlo: **"C:\Mi Directorio\\"**.

Opciones

Las opciones del comando aparecen más abajo divididas en grupos. Los valores se pueden dar sin comillas excepto en estos dos casos: (i) cuando la cadena de valor contiene espacios y (ii) cuando en la descripción de la opción se indique explícitamente que es necesario el uso de comillas. Si una opción toma un valor booleano y no se indica ningún valor, entonces el valor predeterminado de la opción es `TRUE`. Use la opción **--h, --help** para ver información sobre el comando.

▼ Validación y procesamiento

▼ listfile

--listfile = `true|false`
Si el valor es `true`, el argumento *ArchivoEntrada* del comando se entiende como un archivo de

 © 2017-2023 Altova GmbH

texto que contiene un nombre de archivo por línea. Otra opción es enumerar los archivos en la ILC, separados por un espacio. No obstante, recuerde que las ILC tienen un límite de caracteres. Además, no olvide que la opción `--listfile` solamente afecta a los argumentos y no a las opciones.
Valor predeterminado: `false`.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- recurse

  `--recurse =` `true|false`
  Esta opción se utiliza para seleccionar ficheros dentro de subdirectorios (incluso en archivos ZIP). Si el valor es `true`, el argumento *`ArchivoEntrada`* del comando seleccionará el fichero seleccionado también en los subdirectorios. Por ejemplo: `"test.zip|zip\test.xml"` seleccionará los ficheros llamados `test.xml` en todos los subdirectorios de la carpeta ZIP. Si quiere puede usar los caracteres comodín `*` y `?`. Por ejemplo: `*.xml` seleccionaría todos los ficheros de la carpeta ZIP que tengan la extensión `.xml`. Valor predeterminado: `false`
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- Catálogos y recursos globales

  - catalog

    `--catalog =` *`ARCHIVO`*
    Especifica la ruta de acceso absoluta a un archivo de catálogo que no está en el archivo de catálogo raíz instalado. El valor predeterminado es la ruta de acceso absoluta del archivo de catálogo raíz instalado (`<carpeta-instalación>\Altova\RaptorXMLServer2024\etc\RootCatalog.xml`). Consulte el apartado Catálogos XML [48] para obtener más información.

  - user-catalog

    `--user-catalog =` *`ARCHIVO`*
    Especifica la ruta de acceso absoluta a un catálogo XML que debe utilizarse junto con el catálogo raíz. Consulte el apartado Catálogos XML [48] para obtener más información.

  - enable-globalresources

    `--enable-globalresources =` `true|false`
    Habilita la función de recursos globales [55]. Valor predeterminado: `false`.
    Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

  - globalresourceconfig [gc]

    `--gc | --globalresourceconfig =` *`VALOR`*
    Especifica la configuración activa del recurso global [55] (y habilita los recursos globales) [55].

  - globalresourcefile [gr]

    `--gr | --globalresourcefile =` *`ARCHIVO`*
    Especifica el archivo de recursos globales [55] (y habilita los recursos globales) [55].

- Opciones comunes
  - error-format

    --error-format = `text|shortxml|longxml`
    Especifica el formato de la salida de error. Los valores posibles son formatos de texto, XML y XML detallado (`longxml`). Valor predeterminado: `text`.

  - error-limit

    --error-limit = *N | unlimited*
    Especifica el límite de errores con un valor comprendido entre `1` y `9999` o `unlimited` (ilimitado). El valor predeterminado es `100`. Cuando se alcanza el límite de error, se detiene la validación. Esta opción es muy práctica a la hora de limitar el uso del procesador durante la validación o transformación.

  - info-limit

    --info-limit = *N | unlimited*
    Indica el límite del mensaje de información dentro del rango `1-65535` or `unlimited`. Si se alcanza el límite de información indicado, el procesamiento continúa pero ya no se informa de más mensajes. El valor predeterminado es `100`.

  - help

    --help
    Muestra el texto de ayuda para el comando. Por ejemplo `valany --h`. (Otra opción es usar el comando `help` con un argumento. Por ejemplo: `help valany`).

  - listfile

    --listfile = `true|false`
    Si el valor es `true`, el argumento *ArchivoEntrada* del comando se entiende como un archivo de texto que contiene un nombre de archivo por línea. Otra opción es enumerar los archivos en la ILC, separados por un espacio. No obstante, recuerde que las ILC tienen un límite de caracteres. Además, no olvide que la opción `--listfile` solamente afecta a los argumentos y no a las opciones.
    Valor predeterminado: `false`.
    Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

  - log-output

    --log-output = *ARCHIVO*
    Escribe el registro de salida en la URL de archivo indicada. Compruebe que la ILC tiene permiso de escritura en la ubicación de destino.

  - network-timeout

    --network-timeout = *VALOR*
    Indica el tiempo de espera en milisegundos para operaciones remotas de entrada y salida. Valor predeterminado: `40000`.

 © 2017-2023 Altova GmbH

- recurse

  --recurse = `true|false`
  Esta opción se utiliza para seleccionar ficheros dentro de subdirectorios (incluso en archivos ZIP). Si el valor es `true`, el argumento *ArchivoEntrada* del comando seleccionará el fichero seleccionado también en los subdirectorios. Por ejemplo: `"test.zip|zip\test.xml"` seleccionará los ficheros llamados `test.xml` en todos los subdirectorios de la carpeta ZIP. Si quiere puede usar los caracteres comodín `*` y `?`. Por ejemplo: `*.xml` seleccionaría todos los ficheros de la carpeta ZIP que tengan la extensión `.xml`. Valor predeterminado: `false`
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- verbose

  --verbose = `true|false`
  Si el valor es `true`, se genera información adicional durante la validación. Valor predeterminado es `false`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- verbose-output

  --verbose-output = *ARCHIVO*
  Escribe el resultado detallado en el *ARCHIVO* indicado.

- version

  --version
  Muestra el número de versión de RaptorXML Server. Si se utiliza con un comando, escriba la opción `--version` antes del comando.

- warning-limit

  --warning-limit = *N | unlimited*
  Especifica el límite de advertencia en el rango `1-65535` o `unlimited` (ilimitado). El procesamiento continua si se alcanza el límite pero no se registrarán más advertencias. El valor por defecto es `100`.

5.2.3 wfany

El comando **wfany** comprueba si el formato de un documento XML o DTD es correcto según la especificación correspondiente. El tipo de documento se detecta automáticamente.

Windows	**RaptorXML wfany [opciones] *ArchivoEntrada***
Linux	**raptorxml wfany [opciones] *ArchivoEntrada***
Mac	**raptorxml wfany [opciones] *ArchivoEntrada***

El argumento *ArchivoEntrada* es el documento cuyo formato debe comprobarse. Recuerde que como argumento del comando se puede suministrar solamente un documento. El tipo de documento se detecta automáticamente.

Ejemplos

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

Mayúsculas/minúsculas y barras en la línea de comandos

RaptorXML (y **RaptorXMLServer** para comandos administrativos) *en Windows*
raptorxml (y **raptorxmlserver** para comandos administrativos) *en Windows y Unix (Linux, Mac)*

* Las minúsculas (raptorxml y raptorxmlserver) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que las mayúsculas (RaptorXML) funcionan solamente en Windows y Mac.
* Use la barra diagonal en Linux y Mac y la barra diagonal inversa en Windows.

Barra diagonal inversa y espacios en sistemas Windows

En sistemas Windows: si hay espacios o caracteres especiales (por ejemplo en los nombres de archivos o de carpetas, empresas, personas o productos) debe usar comillas: por ejemplo, **"Mi archivo"**. Sin embargo, debe tener en cuenta que una barra diagonal inversa seguida de comillas dobles (por ejemplo, **"C:\Mi Directorio\"**) es posible que no se lea correctamente. Esto se debe a que la barra diagonal inversa también se usa para empezar secuencias de escape y la secuencia de escape **\"** también usa las comillas dobles. Para evitar secuencia de caracteres puede añadir otra barra diagonal inversa: **\\"**. En resumen: si necesita escribir una ruta que contenga espacios y una barra diagonal inversa, esta es la mejor forma de hacerlo: **"C:\Mi Directorio\\"**.

Opciones

Las opciones del comando aparecen más abajo divididas en grupos. Los valores se pueden dar sin comillas excepto en estos dos casos: (i) cuando la cadena de valor contiene espacios y (ii) cuando en la descripción de la opción se indique explícitamente que es necesario el uso de comillas. Si una opción toma un valor booleano y no se indica ningún valor, entonces el valor predeterminado de la opción es TRUE. Use la opción **--h, --help** para ver información sobre el comando.

Validación y procesamiento

listfile

--listfile = true|false
Si el valor es true, el argumento *ArchivoEntrada* del comando se entiende como un archivo de texto que contiene un nombre de archivo por línea. Otra opción es enumerar los archivos en la ILC, separados por un espacio. No obstante, recuerde que las ILC tienen un límite de caracteres. Además, no olvide que la opción --listfile solamente afecta a los argumentos y no a las opciones.
Valor predeterminado: false.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en true.

 © 2017-2023 Altova GmbH

- Catálogos y recursos globales
  - catalog

    --catalog = *ARCHIVO*
    Especifica la ruta de acceso absoluta a un archivo de catálogo que no está en el archivo de catálogo raíz instalado. El valor predeterminado es la ruta de acceso absoluta del archivo de catálogo raíz instalado (`<carpeta-instalación>\Altova\RaptorXMLServer2024\etc\RootCatalog.xml`). Consulte el apartado Catálogos XML [48] para obtener más información.

  - user-catalog

    --user-catalog = *ARCHIVO*
    Especifica la ruta de acceso absoluta a un catálogo XML que debe utilizarse junto con el catálogo raíz. Consulte el apartado Catálogos XML [48] para obtener más información.

  - enable-globalresources

    --enable-globalresources = `true|false`
    Habilita la función de recursos globales [55]. Valor predeterminado: `false`.
    Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

  - globalresourceconfig [gc]

    --gc | --globalresourceconfig = *VALOR*
    Especifica la configuración activa del recurso global [55] (y habilita los recursos globales) [55].

  - globalresourcefile [gr]

    --gr | --globalresourcefile = *ARCHIVO*
    Especifica el archivo de recursos globales [55] (y habilita los recursos globales) [55].

- Opciones comunes
  - error-format

    --error-format = `text|shortxml|longxml`
    Especifica el formato de la salida de error. Los valores posibles son formatos de texto, XML y XML detallado (`longxml`). Valor predeterminado: `text`.

  - error-limit

    --error-limit = *N | unlimited*
    Especifica el límite de errores con un valor comprendido entre `1` y `9999` o `unlimited` (ilimitado). El valor predeterminado es `100`. Cuando se alcanza el límite de error, se detiene la validación. Esta opción es muy práctica a la hora de limitar el uso del procesador durante la validación o transformación.

  - info-limit

--info-limit = *N | unlimited*
Indica el límite del mensaje de información dentro del rango `1-65535` or `unlimited`. Si se alcanza el límite de información indicado, el procesamiento continúa pero ya no se informa de más mensajes. El valor predeterminado es `100`.

- help

--help
Muestra el texto de ayuda para el comando. Por ejemplo `valany --h`. (Otra opción es usar el comando `help` con un argumento. Por ejemplo: `help valany`).

- listfile

--listfile = `true|false`
Si el valor es `true`, el argumento *ArchivoEntrada* del comando se entiende como un archivo de texto que contiene un nombre de archivo por línea. Otra opción es enumerar los archivos en la ILC, separados por un espacio. No obstante, recuerde que las ILC tienen un límite de caracteres. Además, no olvide que la opción `--listfile` solamente afecta a los argumentos y no a las opciones.
Valor predeterminado: `false`.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- log-output

--log-output = *ARCHIVO*
Escribe el registro de salida en la URL de archivo indicada. Compruebe que la ILC tiene permiso de escritura en la ubicación de destino.

- network-timeout

--network-timeout = *VALOR*
Indica el tiempo de espera en milisegundos para operaciones remotas de entrada y salida. Valor predeterminado: `40000`.

- recurse

--recurse = `true|false`
Esta opción se utiliza para seleccionar ficheros dentro de subdirectorios (incluso en archivos ZIP). Si el valor es `true`, el argumento *ArchivoEntrada* del comando seleccionará el fichero seleccionado también en los subdirectorios. Por ejemplo: `"test.zip|zip\test.xml"` seleccionará los ficheros llamados `test.xml` en todos los subdirectorios de la carpeta ZIP. Si quiere puede usar los caracteres comodín `*` y `?`. Por ejemplo: `*.xml` seleccionaría todos los ficheros de la carpeta ZIP que tengan la extensión `.xml`. Valor predeterminado: `false`
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- verbose

--verbose = `true|false`
Si el valor es `true`, se genera información adicional durante la validación. Valor predeterminado es `false`.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en

 © 2017-2023 Altova GmbH

`true`.

- verbose-output

  `--verbose-output =` *`ARCHIVO`*
  Escribe el resultado detallado en el *`ARCHIVO`* indicado.

- version

  `--version`
  Muestra el número de versión de RaptorXML Server. Si se utiliza con un comando, escriba la opción `--version` antes del comando.

- warning-limit

  `--warning-limit =` *`N | unlimited`*
  Especifica el límite de advertencia en el rango `1-65535` o `unlimited` (ilimitado). El procesamiento continua si se alcanza el límite pero no se registrarán más advertencias. El valor por defecto es `100`.

5.3 Comandos XQuery

Estos son los comandos XQuery:

xquery 96	Para ejecutar documentos XQuery y, si se quiere, con un documento de entrada.
xqueryupdate 96	Para ejecutar una actualización XQuery con un documento XQuery y el documento XML de entrada que se debe actualizar.
valxquery 114	Para validar documentos XQuery.
valxqueryupdate 121	Para validar documentos XQuery de actualización.

5.3.1 xquery

El comando **xquery** toma un archivo XQuery como único argumento y lo ejecuta con un archivo de entrada opcional para generar un archivo de salida. Los archivos de entrada y salida se especifican como opciones.

Windows	**RaptorXML xquery [opciones] *Archivo-XQuery***
Linux	**raptorxml xquery [opciones] *Archivo-XQuery***
Mac	**raptorxml xquery [opciones] *Archivo-XQuery***

El argumento *Archivo-XQuery* es la ruta de acceso y el nombre del archivo XQuery que debe ejecutarse. Puede usar tanto XQuery 1.0 como 3.0, pero la versión predeterminada es 3.0.

Ejemplos

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

▼ Mayúsculas/minúsculas y barras en la línea de comandos

RaptorXML (y **RaptorXMLServer** para comandos administrativos) *en Windows*
raptorxml (y **raptorxmlserver** para comandos administrativos) *en Windows y Unix (Linux, Mac)*

* Las minúsculas (raptorxml y raptorxmlserver) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que las mayúsculas (RaptorXML) funcionan solamente en Windows y Mac.
* Use la barra diagonal en Linux y Mac y la barra diagonal inversa en Windows.

▼ Barra diagonal inversa y espacios en sistemas Windows

 © 2017-2023 Altova GmbH

En sistemas Windows: si hay espacios o caracteres especiales (por ejemplo en los nombres de archivos o de carpetas, empresas, personas o productos) debe usar comillas: por ejemplo, **"Mi archivo"**. Sin embargo, debe tener en cuenta que una barra diagonal inversa seguida de comillas dobles (por ejemplo, **"C:\Mi Directorio\"**) es posible que no se lea correctamente. Esto se debe a que la barra diagonal inversa también se usa para empezar secuencias de escape y la secuencia de escape **\"** también usa las comillas dobles. Para evitar secuencia de caracteres puede añadir otra barra diagonal inversa: **\\"**. En resumen: si necesita escribir una ruta que contenga espacios y una barra diagonal inversa, esta es la mejor forma de hacerlo: **"C:\Mi Directorio\\"**.

Opciones

Las opciones del comando aparecen más abajo divididas en grupos. Los valores se pueden dar sin comillas excepto en estos dos casos: (i) cuando la cadena de valor contiene espacios y (ii) cuando en la descripción de la opción se indique explícitamente que es necesario el uso de comillas. Si una opción toma un valor booleano y no se indica ningún valor, entonces el valor predeterminado de la opción es `TRUE`. Use la opción **--h, --help** para ver información sobre el comando.

- Procesamiento XQuery
  - indent-characters

    --indent-characters = *VALOR*
    Especifica la cadena de caracteres que debe usarse como sangría.

  - input

    --input = *ARCHIVO*
    La URL del archivo XML que se debe transformar.

  - omit-xml-declaration

    --omit-xml-declaration = `true|false`
    Opción de serialización que especifica si la declaración XML se omite en el resultado o no. Si el valor es `true`, el documento de salida no tendrá una declaración XML. Si el valor es `false`, se incluye una declaración XML en el documento de salida. Valor predeterminado: `false`.
    Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

  - output, xsltoutput

    output = *ARCHIVO*, **xsltoutput =** *ARCHIVO*
    La URL del archivo de salida principal. Por ejemplo, en caso de tener varios archivos HTML de salida, el archivo de salida principal será la ubicación del archivo HTML del punto de entrada. Los demás archivos de salida (como archivos de imagen generados) se indican con `xslt-additional-output-files`. Si no se especifica la opción `--output` ni la opción `--xsloutput`, se genera un resultado estándar.

  - output-encoding

    --output-encoding = *VALOR*
    El valor del atributo `encoding` del documento de salida. Son valores válidos todos los nombres del registro de juego de caracteres IANA. Valor predeterminado: `UTF-8`.

- output-indent

  --output-indent = `true|false`

  Si el valor es `true`, la sangría del documento de salida seguirá su estructura jerárquica. Si el valor es `false`, el documento de salida no tendrá sangría jerárquica. Valor predeterminado: `false`.

  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- output-method

  --output-method = `xml|html|xhtml|text`

  Especifica el formato de salida. Valor predeterminado: `xml`.

- param [p]

  --p | --param = *CLAVE:VALOR*

  - ***XQuery***

    Especifica el valor de un parámetro externo. En el documento XQuery los parámetros externos se declaran con la declaración `declare variable` seguida de un nombre de variable y después la palabra clave `external` seguida del punto y coma final. Por ejemplo: `declare variable $foo as xs:string external;`

    Al usar la palabra clave `external`, `$foo` se convierte en parámetro externo y su valor se pasa en tiempo de ejecución desde una fuente externa. El parámetro externo recibe un valor con el comando de la ILC. Por ejemplo: `--param=foo:'MiNombre'`

    En la descripción anterior, *CLAVE* es el nombre de parámetro externo y *VALOR* es su valor, dado como expresión XPath. Los nombres de parámetro utilizados en la ILC deben declararse en el documento XQuery. Si se pasan valores a varios parámetros externos en la ILC, cada parámetro debe llevar una opción `--param` distinta. Si la expresión XPath contiene espacios, entonces debe estar entre comillas dobles.

  - ***XSLT***

    Especifica un parámetro global de la hoja de estilos. *CLAVE* es el nombre del parámetro y *VALOR* es una expresión XPath que da un valor al parámetro. Los nombres de parámetro utilizados en la ILC deben declararse en la hoja de estilos. Si usa más de un parámetro, debe usar el modificador `--param` antes de cada parámetro. Si la expresión XPath incluye espacios, entonces debe ir entre comillas dobles, tanto si el espacio está en la expresión propiamente dicha o en un literal de cadena de la expresión. Por ejemplo:

    ```
    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|falsetrue|false`

  Si se establece en `true` reduce a advertencia cualquier error que se detecte en el contexto estático XPath. Mientras que un error provocaría un error, una advertencia permite que el procesamiento continúe. El valor predeterminado es `false`.

 © 2017-2023 Altova GmbH

- xquery-version

  --xquery-version = 1|1.0|3|3.0|3.1
  Indica si el procesador XQuery debe usar XQuery 1.0 o 3.0. Valor predeterminado: 3.1

- Instancias y esquemas XML
  - load-xml-with-psvi

    --load-xml-with-psvi = true|false
    Habilita la validación de archivos XML de entrada y genera información posterior a la validación. Valor predeterminado: true.

  - schema-imports

    --schema-imports = load-by-schemalocation | load-preferring-schemalocation | load-by-namespace | load-combining-both | license-namespace-only
    Esta opción indica el comportamiento de los elementos xs:import. Cada uno de estos elementos tiene un atributo opcional namespace y un atributo opcional schemaLocation: <import namespace="unEspacioNombres" schemaLocation="unaURL">. La opción indica si se debe cargar un documento de esquema o solo autorizar a un espacio de nombres. Si la opción indica que se debe cargar un documento de esquema, entonces indica también qué información debe utilizarse para encontrar el documento de esquema. Valor predeterminado: load-preferring-schemalocation.

    - load-by-schemalocation: el valor del atributo schemaLocation se utiliza para buscar el esquema, teniendo en cuenta las asignaciones de catálogo[48]. Si está presente el atributo namespace, se importa el espacio de nombres (con licencia).
    - load-preferring-schemalocation: si está presente, se utiliza el atributo schemaLocation teniendo en cuenta las asignaciones de catálogo[48]. Si no está presente el atributo schemaLocation, entonces se usa el valor del atributo namespace a través de las asignaciones de catálogo[48]. Este es el **valor predeterminado**.
    - load-by-namespace: el valor del atributo namespace se utiliza para buscar el esquema por medio de una asignación de catálogo[48].
    - load-combining-both: si el atributo namespace o schemaLocation tiene una asignación de catálogo[48], entonces se usa la asignación. Si ambos atributos tienen asignaciones de catálogo[48], entonces es el valor de la opción --schema-mapping (opción XML/XSD[240]) decide qué asignación se utiliza. Si no hay ninguna asignación de catálogo[48], entonces se usa el atributo schemaLocation.
    - license-namespace-only: se importa el espacio de nombres. No se importa el documento de esquema.

  - schemalocation-hints

    --schemalocation-hints = load-by-schemalocation | load-by-namespace | load-combining-both | ignore
    Determina el comportamiento predeterminado de los atributos xsi:schemaLocation y xsi:noNamespaceSchemaLocation. Indica si se debe cargar un documento de esquema y, si así es, indica qué información debe utilizarse para encontrarlo. Valor predeterminado: load-by-schemalocation.

    - **Valor predeterminado**: load-by-schemalocation. Este valor toma la URL de la ubicación del

esquema[403] de los atributos `xsi:schemaLocation` y `xsi:noNamespaceSchemaLocation` de los documentos de instancia XML.

- El valor `load-by-namespace` toma la parte de espacio de nombres[403] del atributo `xsi:schemaLocation` y una cadena vacía en el caso del atributo `xsi:noNamespaceSchemaLocation` y encuentra el esquema por medio de una asignación de catálogo[48].
- Si usa el valor `load-combining-both` y el espacio de nombres o la URL tienen una asignación de catálogo[48], se usa dicha asignación. Si ambos tienen asignaciones de catálogo[48], el valor de la opción `schema-mapping` (opción XML/XSD[240]) decide qué asignación se utiliza. Si ni el espacio de nombres ni la URL tiene una asignación de catálogo, se usa la URL.
- El valor `ignore` ignora los atributos `xsi:schemaLocation` y `xsi:noNamespaceSchemaLocation`.

▼ schema-mapping

`--schema-mapping =` `prefer-schemalocation | prefer-namespace`
Si se usa la ubicación y el espacio de nombres para buscar el documento de esquema, esta opción indica cuál de ellos debe ser la opción preferida durante la búsqueda en el catálogo.
Si la opción `--schemalocation-hints` o la opción `--schema-imports` tiene el valor `load-combining-both` y si las partes de espacio de nombres y URL pertinentes tienen asignaciones de catálogo[48], entonces el valor de la opción especifica cuál de las dos asignaciones se utiliza (la asignación del espacio de nombres o de la URL: el valor `prefer-schemalocation` se refiere a la asignación de la URL). Valor predeterminado: `prefer-schemalocation`.

▼ xinclude

`--xinclude =` `true|false`
Habilita la compatibilidad con inclusiones XML (XInclude). Si el valor es `false`, los elementos XInclude `include` se ignoran. Valor predeterminado: `false`.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

▼ xml-mode

`--xml-mode =` `wf|id|valid`
Especifica el modo de procesamiento XML que debe utilizarse para el documento de instancia XML: `wf`=comprobación de formato; `id`=comprobación de formato con ID/IDREF; `valid`=validación. Valor predeterminado: `wf`. Recuerde que el valor `valid` exige que cada documento de instancia que se cargue durante el procesamiento haga referencia a una DTD. Si no existe ninguna DTD, se generará un error.

▼ xml-mode-for-schemas

`--xml-mode-for-schemas =` `wf|id|valid`
Especifica el modo de procesamiento XML que debe utilizarse para el documento de instancia XML: `wf`=comprobación de formato; `id`=comprobación de formato con ID/IDREF; `valid`=validación. Valor predeterminado: `wf`. Recuerde que el valor `valid` exige que cada documento de esquema que se cargue durante el procesamiento haga referencia a una DTD. Si no existe ninguna DTD, se generará un error.

▼ xml-validation-error-as-warning

`--xml-validation-error-as-warning =` `true|false`

 © 2017-2023 Altova GmbH

Si es `true`, tratar los errores de validación como advertencias. Si los errores se tratan como advertencias, el procesamiento adicional como la transformación XSLT, proseguirá independientemente de los errores. Por defecto es `false`.

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

  `--xpath-static-type-errors-as-warnings =` `true|falsetrue|false`
  Si se establece en `true` reduce a advertencia cualquier error que se detecte en el contexto estático XPath. Mientras que un error provocaría un error, una advertencia permite que el procesamiento continúe. El valor predeterminado es `false`.

- xsd

  `--xsd =` *ARCHIVO*
  Especifica qué esquemas XML deben utilizarse para la validación de documentos XML. Si quiere especificar más de un esquema, añada la opción varias veces.

- xsd-version

  `--xsd-version =` `1.0|1.1|detect`

  Especifica qué versión de la especificación Schema Definition Language (XSD) del W3C se debe usar. Valor predeterminado: `1.0`.

  Esta opción también puede ser útil si quiere ver en qué aspectos no es compatible un esquema 1.0 con la especificación 1.1. El valor `detect` es una característica de Altova. Permite detectar la versión del esquema XML (`1.0` o `1.1`) leyendo el valor del atributo `vc:minVersion` del elemento `<xs:schema>` del documento. Si el valor del atributo `@vc:minVersion` es `1.1`, se entiende que la versión del esquema es `1.1`. Si el atributo tiene otro valor que no sea `1.1` (o si no está presente el atributo `@vc:minVersion`), se entiende que la versión del esquema es `1.0`.

- Catálogos y recursos globales
  - catalog

    `--catalog =` *ARCHIVO*
    Especifica la ruta de acceso absoluta a un archivo de catálogo que no está en el archivo de catálogo raíz instalado. El valor predeterminado es la ruta de acceso absoluta del archivo de catálogo raíz instalado (`<carpeta-instalación>\Altova\RaptorXMLServer2024\etc\RootCatalog.xml`). Consulte el apartado Catálogos XML 48 para obtener más información.

  - user-catalog

    `--user-catalog =` *ARCHIVO*
    Especifica la ruta de acceso absoluta a un catálogo XML que debe utilizarse junto con el catálogo raíz. Consulte el apartado Catálogos XML 48 para obtener más información.

  - enable-globalresources

    `--enable-globalresources =` `true|false`
    Habilita la función de recursos globales 55. Valor predeterminado: `false`.
    Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en

`true`.

- globalresourceconfig [gc]

  `--gc | --globalresourceconfig =` *`VALOR`*
  Especifica la configuración activa del recurso global[55] (y habilita los recursos globales)[55].

- globalresourcefile [gr]

  `--gr | --globalresourcefile =` *`ARCHIVO`*
  Especifica el archivo de recursos globales[55] (y habilita los recursos globales)[55].

- Extensiones

  Estas opciones definen cómo se gestionan las funciones de extensión especiales disponibles en la edición Enterprise Edition de varios productos de Altova (como XMLSpy Enterprise Edition). Su uso se describe detalladamente en el manual del usuario de cada producto.

  - chartext-disable

    `--chartext-disable =` `true|false`
    Deshabilita las extensiones de gráficos. Valor predeterminado: `false`.
    Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

  - dotnetext-disable

    `--dotnetext-disable =` `true|false`
    Deshabilita las extensiones .NET. Valor predeterminado: `false`.
    Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

  - jvm-location

    `--jvm-location =` *`ARCHIVO`*
    `ARCHIVO` especifica la ubicación del equipo virtual Java (*DLL* en Windows, objeto *compartido (shared object)* en Linux. El JVM se necesita si usa las funciones de extensión Java[512] en su código XSLT/XQuery. El valor predeterminado es `false`.

  - javaext-barcode-location

    `--javaext-barcode-location =` *`ARCHIVO`*
    Especifica la ruta de acceso de la carpeta que contiene el archivo de extensión de código de barras `AltovaBarcodeExtension.jar`. La ruta de acceso debe darse en uno de estos formatos:

    - Un URI de archivo (ejemplo: `--javaext-barcode-location="file:///C:/Archivos de programa/Altova/RaptorXMLServer2024/etc/jar/"`)
    - Una ruta de acceso Windows con caracteres de escape para las barras diagonales inversas (ejemplo: `--javaext-barcode-location="C:\\Archivos de programa\\Altova\ \RaptorXMLServer2024\\etc\\jar\\"`)

  - javaext-disable

    `--javaext-disable =` `true|false`

 © 2017-2023 Altova GmbH

Deshabilita las extensiones Java. Valor predeterminado: `false`.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- Opciones comunes
  - error-format

    `--error-format =` `text|shortxml|longxml`
    Especifica el formato de la salida de error. Los valores posibles son formatos de texto, XML y XML detallado (`longxml`). Valor predeterminado: `text`.

  - error-limit

    `--error-limit =` *`N | unlimited`*
    Especifica el límite de errores con un valor comprendido entre `1` y `9999` o `unlimited` (ilimitado). El valor predeterminado es `100`. Cuando se alcanza el límite de error, se detiene la validación. Esta opción es muy práctica a la hora de limitar el uso del procesador durante la validación o transformación.

  - info-limit

    `--info-limit =` *`N | unlimited`*
    Indica el límite del mensaje de información dentro del rango `1-65535` or `unlimited`. Si se alcanza el límite de información indicado, el procesamiento continúa pero ya no se informa de más mensajes. El valor predeterminado es `100`.

  - help

    `--help`
    Muestra el texto de ayuda para el comando. Por ejemplo `valany --h`. (Otra opción es usar el comando `help` con un argumento. Por ejemplo: `help valany`).

  - listfile

    `--listfile =` `true|false`
    Si el valor es `true`, el argumento *`ArchivoEntrada`* del comando se entiende como un archivo de texto que contiene un nombre de archivo por línea. Otra opción es enumerar los archivos en la ILC, separados por un espacio. No obstante, recuerde que las ILC tienen un límite de caracteres. Además, no olvide que la opción `--listfile` solamente afecta a los argumentos y no a las opciones.
    Valor predeterminado: `false`.
    Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

  - log-output

    `--log-output =` *`ARCHIVO`*
    Escribe el registro de salida en la URL de archivo indicada. Compruebe que la ILC tiene permiso de escritura en la ubicación de destino.

  - network-timeout

--network-timeout = *VALOR*
Indica el tiempo de espera en milisegundos para operaciones remotas de entrada y salida. Valor predeterminado: `40000`.

- recurse

  --recurse = `true|false`
  Esta opción se utiliza para seleccionar ficheros dentro de subdirectorios (incluso en archivos ZIP). Si el valor es `true`, el argumento *ArchivoEntrada* del comando seleccionará el fichero seleccionado también en los subdirectorios. Por ejemplo: `"test.zip|zip\test.xml"` seleccionará los ficheros llamados `test.xml` en todos los subdirectorios de la carpeta ZIP. Si quiere puede usar los caracteres comodín `*` y `?`. Por ejemplo: `*.xml` seleccionaría todos los ficheros de la carpeta ZIP que tengan la extensión `.xml`. Valor predeterminado: `false`
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- verbose

  --verbose = `true|false`
  Si el valor es `true`, se genera información adicional durante la validación. Valor predeterminado es `false`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- verbose-output

  --verbose-output = *ARCHIVO*
  Escribe el resultado detallado en el *ARCHIVO* indicado.

- version

  --version
  Muestra el número de versión de RaptorXML Server. Si se utiliza con un comando, escriba la opción `--version` antes del comando.

- warning-limit

  --warning-limit = *N | unlimited*
  Especifica el límite de advertencia en el rango `1-65535` o `unlimited` (ilimitado). El procesamiento continua si se alcanza el límite pero no se registrarán más advertencias. El valor por defecto es `100`.

5.3.2 xqueryupdate

El comando **xqueryupdate** toma un archivo XQuery o xQuery Update como único argumento y lo ejecuta. Si se indica un archivo XML de entrada opcional, entonces este archivo XML se procesa con los comandos XQuery Update indicados en ***Archivo-XQuery-actualizado***. En este caso, las actualizaciones se pueden aplicar directamente al archivo de entrada de actualizado o los datos XML se pueden escribir en un archivo XML de salida. Los archivos de entrada y salida se especifican en las opciones del comando. Si ***Archivo-***

 © 2017-2023 Altova GmbH

XQuery-actualizado solo contiene instrucciones XQuery y no XQuery Update, entonces el comando ejecuta XQuery directamente.

Windows	**RaptorXML xqueryupdate [opciones]** ***Archivo-XQuery-actualizado***
Linux	**raptorxml xqueryupdate [opciones]** ***Archivo-XQuery-actualizado***
Mac	**raptorxml xqueryupdate [opciones]** ***Archivo-XQuery-actualizado***

- El argumento Archivo-XQuery-actualizado es la ruta de acceso y el nombre del archivo XQuery (.xq) o XQuery Update (.xqu) que se debe ejecutar. Si el archivo contiene instrucciones XQuery Update, entonces estas se ejecutan en el archivo XML. De lo contrario, el comando funciona como un comando de ejecución XQuery.
- También puede especificar si se utiliza XQuery Update 1.0 o 3.0 (la opción predeterminada XQuery Update 3.0).

Ejemplos

- **raptorxml** `xqueryupdate --output=c:\Salida.xml c:\TestQuery.xq` (Escribe la salida del archivo XQuery en el archivo de salida.)
- **raptorxml** `xqueryupdate --input=c:\Entrada.xml --output=c:\Salida.xml --param=company:"Altova" --p=date:"2006-01-01" c:\TestQuery.xq` (Actualiza `Input.xml` con las instrucciones de actualización de `UpdateFile.xqu` y escribe la actualización en `Output.xml`.)
- **raptorxml** `xqueryupdate --input=c:\Entrada.xml --output=c:\Salida.xml --param=source:" doc( 'c:\test\books.xml' )//book " c:\TestQuery.xq` (Actualiza `Input.xml` con las instrucciones de actualización de `UpdateFile.xq`. No se crea el archivo `Output.xml`.)
- **raptorxml** `xqueryupdate --output=c:\Salida.xml --omit-xml-declaration=false --output-encoding=ASCII c:\TestQuery.xq` (Las actualizaciones se descartan. El archivo de entrada no se modifica. El archivo `Output.xml` se crea pero no contiene XML actualizado.)
- **raptorxml** `xqueryupdate --input=c:\Input.xml --output=c:\Salida.xml c:\TestQuery.xqu` (Las actualizaciones se descartan, como en el ejemplo anterior. Esto es debido a que el valor predeterminado de la opción `--updated-xml` es `discard`.)

▼ Mayúsculas/minúsculas y barras en la línea de comandos

RaptorXML (y **RaptorXMLServer** para comandos administrativos) *en Windows*
raptorxml (y **raptorxmlserver** para comandos administrativos) *en Windows y Unix (Linux, Mac)*

* Las minúsculas (`raptorxml` y `raptorxmlserver`) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que las mayúsculas (`RaptorXML`) funcionan solamente en Windows y Mac.
* Use la barra diagonal en Linux y Mac y la barra diagonal inversa en Windows.

▼ Barra diagonal inversa y espacios en sistemas Windows

En sistemas Windows: si hay espacios o caracteres especiales (por ejemplo en los nombres de archivos o de carpetas, empresas, personas o productos) debe usar comillas: por ejemplo, **"Mi archivo"**. Sin embargo, debe tener en cuenta que una barra diagonal inversa seguida de comillas dobles (por ejemplo, **"C:\Mi Directorio\"**) es posible que no se lea correctamente. Esto se debe a que la barra diagonal inversa también se usa para empezar secuencias de escape y la secuencia de escape **\"** también usa las comillas dobles. Para evitar secuencia de caracteres puede añadir otra barra diagonal inversa: **\\"**. En

resumen: si necesita escribir una ruta que contenga espacios y una barra diagonal inversa, esta es la mejor forma de hacerlo: `"C:\Mi Directorio\\"`.

Opciones

Las opciones del comando aparecen más abajo divididas en grupos. Los valores se pueden dar sin comillas excepto en estos dos casos: (i) cuando la cadena de valor contiene espacios y (ii) cuando en la descripción de la opción se indique explícitamente que es necesario el uso de comillas. Si una opción toma un valor booleano y no se indica ningún valor, entonces el valor predeterminado de la opción es `TRUE`. Use la opción **`--h, --help`** para ver información sobre el comando.

- Procesamiento XQuery Update
  - Procesamiento XQuery
    - indent-characters

      `--indent-characters =` *VALOR*
      Especifica la cadena de caracteres que debe usarse como sangría.

    - input

      `--input =` *ARCHIVO*
      La URL del archivo XML que se debe transformar.

    - omit-xml-declaration

      `--omit-xml-declaration =` `true|false`
      Opción de serialización que especifica si la declaración XML se omite en el resultado o no. Si el valor es `true`, el documento de salida no tendrá una declaración XML. Si el valor es `false`, se incluye una declaración XML en el documento de salida. Valor predeterminado: `false`.
      Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

    - output, xsltoutput

      `output =` *ARCHIVO,* **`xsltoutput =`** *ARCHIVO*
      La URL del archivo de salida principal. Por ejemplo, en caso de tener varios archivos HTML de salida, el archivo de salida principal será la ubicación del archivo HTML del punto de entrada. Los demás archivos de salida (como archivos de imagen generados) se indican con `xslt-additional-output-files`. Si no se especifica la opción `--output` ni la opción `--xsloutput`, se genera un resultado estándar.

    - output-encoding

      `--output-encoding =` *VALOR*
      El valor del atributo `encoding` del documento de salida. Son valores válidos todos los nombres del registro de juego de caracteres IANA. Valor predeterminado: `UTF-8`.

    - output-indent

      `--output-indent =` `true|false`

 © 2017-2023 Altova GmbH

Si el valor es `true`, la sangría del documento de salida seguirá su estructura jerárquica. Si el valor es `false`, el documento de salida no tendrá sangría jerárquica. Valor predeterminado: `false`.

Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

▼ output-method

`--output-method =` `xml|html|xhtml|text`

Especifica el formato de salida. Valor predeterminado: `xml`.

▼ param [p]

`--p | --param =` *`CLAVE:VALOR`*

▣ ***XQuery***

Especifica el valor de un parámetro externo. En el documento XQuery los parámetros externos se declaran con la declaración `declare variable` seguida de un nombre de variable y después la palabra clave `external` seguida del punto y coma final. Por ejemplo: `declare variable $foo as xs:string external;`

Al usar la palabra clave `external`, `$foo` se convierte en parámetro externo y su valor se pasa en tiempo de ejecución desde una fuente externa. El parámetro externo recibe un valor con el comando de la ILC. Por ejemplo: `--param=foo:'MiNombre'`

En la descripción anterior, *`CLAVE`* es el nombre de parámetro externo y *`VALOR`* es su valor, dado como expresión XPath. Los nombres de parámetro utilizados en la ILC deben declararse en el documento XQuery. Si se pasan valores a varios parámetros externos en la ILC, cada parámetro debe llevar una opción `--param` distinta. Si la expresión XPath contiene espacios, entonces debe estar entre comillas dobles.

▣ ***XSLT***

Especifica un parámetro global de la hoja de estilos. *`CLAVE`* es el nombre del parámetro y *`VALOR`* es una expresión XPath que da un valor al parámetro. Los nombres de parámetro utilizados en la ILC deben declararse en la hoja de estilos. Si usa más de un parámetro, debe usar el modificador `--param` antes de cada parámetro. Si la expresión XPath incluye espacios, entonces debe ir entre comillas dobles, tanto si el espacio está en la expresión propiamente dicha o en un literal de cadena de la expresión. Por ejemplo:

```
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|falsetrue|false`

Si se establece en `true` reduce a advertencia cualquier error que se detecte en el contexto estático XPath. Mientras que un error provocaría un error, una advertencia permite que el procesamiento continúe. El valor predeterminado es `false`.

▼ xquery-version

--xquery-version = 1|1.0|3|3.0|3.1
Indica si el procesador XQuery debe usar XQuery 1.0 o 3.0. Valor predeterminado: 3.1

- xquery-update-version

  --xquery-update-version = 1|1.0|3|3.0|
  Indica si el procesador XQuery debería usar XQuery Update Facility 1.0 o XQuery Update Facility 3.0. Valor predeterminado: 3.

- keep-formatting

  --keep-formatting = true|false
  Conserva en la medida de lo posible el formato del documento de destino. Valor predeterminado: true.

- updated-xml

  --updated-xml = discard|writeback|asmainresult
  Indica qué se debe hacer con el archivo XML actualizado.

  - discard: las actualizaciones se descartan y no se escriben en el archivo. Ni el archivo de entrada ni el de salida se actualizan. Este es el valor predeterminado.
  - writeback: escribe las actualizaciones en el archivo XML de entrada indicado en la opción --input.
  - asmainresult: escribe las actualizaciones en el archivo XML de salida indicado en la opción --output. Si no se indicó la opción --output, las actualizaciones se escriben en el archivo estándar de salida. En ambos casos el archivo XML no se modifica.

  Valor predeterminado: discard.

- Instancias y esquemas XML

  - load-xml-with-psvi

    --load-xml-with-psvi = true|false
    Habilita la validación de archivos XML de entrada y genera información posterior a la validación. Valor predeterminado: true.

  - xinclude

    --xinclude = true|false
    Habilita la compatibilidad con inclusiones XML (XInclude). Si el valor es false, los elementos XInclude include se ignoran. Valor predeterminado: false.
    Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en true.

  - schema-imports

    --schema-imports = load-by-schemalocation | load-preferring-schemalocation | load-by-namespace | load-combining-both | license-namespace-only
    Esta opción indica el comportamiento de los elementos xs:import. Cada uno de estos elementos tiene un atributo opcional namespace y un atributo opcional schemaLocation: <import

 © 2017-2023 Altova GmbH

`namespace="unEspacioNombres" schemaLocation="unaURL">`. La opción indica si se debe cargar un documento de esquema o solo autorizar a un espacio de nombres. Si la opción indica que se debe cargar un documento de esquema, entonces indica también qué información debe utilizarse para encontrar el documento de esquema. Valor predeterminado: `load-preferring-schemalocation`.

- `load-by-schemalocation`: el valor del atributo `schemaLocation` se utiliza para buscar el esquema, teniendo en cuenta las asignaciones de catálogo[48]. Si está presente el atributo `namespace`, se importa el espacio de nombres (con licencia).
- `load-preferring-schemalocation`: si está presente, se utiliza el atributo `schemaLocation` teniendo en cuenta las asignaciones de catálogo[48]. Si no está presente el atributo `schemaLocation`, entonces se usa el valor del atributo `namespace` a través de las asignaciones de catálogo[48]. Este es el **valor predeterminado**.
- `load-by-namespace`: el valor del atributo `namespace` se utiliza para buscar el esquema por medio de una asignación de catálogo[48].
- `load-combining-both`: si el atributo `namespace` o `schemaLocation` tiene una asignación de catálogo[48], entonces se usa la asignación. Si ambos atributos tienen asignaciones de catálogo[48], entonces es el valor de la opción `--schema-mapping` (opción XML/XSD[240]) decide qué asignación se utiliza. Si no hay ninguna asignación de catálogo[48], entonces se usa el atributo `schemaLocation`.
- `license-namespace-only`: se importa el espacio de nombres. No se importa el documento de esquema.

▼ schemalocation-hints

`--schemalocation-hints =` `load-by-schemalocation | load-by-namespace | load-combining-both | ignore`

Determina el comportamiento predeterminado de los atributos `xsi:schemaLocation` y `xsi:noNamespaceSchemaLocation`. Indica si se debe cargar un documento de esquema y, si así es, indica qué información debe utilizarse para encontrarlo. Valor predeterminado: `load-by-schemalocation`.

- **Valor predeterminado**: `load-by-schemalocation`. Este valor toma la URL de la ubicación del esquema[403] de los atributos `xsi:schemaLocation` y `xsi:noNamespaceSchemaLocation` de los documentos de instancia XML.
- El valor `load-by-namespace` toma la parte de espacio de nombres[403] del atributo `xsi:schemaLocation` y una cadena vacía en el caso del atributo `xsi:noNamespaceSchemaLocation` y encuentra el esquema por medio de una asignación de catálogo[48].
- Si usa el valor `load-combining-both` y el espacio de nombres o la URL tienen una asignación de catálogo[48], se usa dicha asignación. Si ambos tienen asignaciones de catálogo[48], el valor de la opción `schema-mapping` (opción XML/XSD[240]) decide qué asignación se utiliza. Si ni el espacio de nombres ni la URL tiene una asignación de catálogo, se usa la URL.
- El valor `ignore` ignora los atributos `xsi:schemaLocation` y `xsi:noNamespaceSchemaLocation`.

▼ schema-mapping

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

Si se usa la ubicación y el espacio de nombres para buscar el documento de esquema, esta opción indica cuál de ellos debe ser la opción preferida durante la búsqueda en el catálogo.

Si la opción `--schemalocation-hints` o la opción `--schema-imports` tiene el valor `load-combining-both` y si las partes de espacio de nombres y URL pertinentes tienen asignaciones de catálogo[48], entonces el valor de la opción especifica cuál de las dos asignaciones se utiliza (la asignación del espacio de nombres o de la URL: el valor `prefer-schemalocation` se refiere a la asignación de la URL). Valor predeterminado: `prefer-schemalocation`.

- xinclude

  `--xinclude =` `true|false`
  Habilita la compatibilidad con inclusiones XML (XInclude). Si el valor es `false`, los elementos XInclude `include` se ignoran. Valor predeterminado: `false`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- xml-mode

  `--xml-mode =` `wf|id|valid`
  Especifica el modo de procesamiento XML que debe utilizarse para el documento de instancia XML: `wf`=comprobación de formato; `id`=comprobación de formato con ID/IDREF; `valid`=validación. Valor predeterminado: `wf`. Recuerde que el valor `valid` exige que cada documento de instancia que se cargue durante el procesamiento haga referencia a una DTD. Si no existe ninguna DTD, se generará un error.

- xml-mode-for-schemas

  `--xml-mode-for-schemas =` `wf|id|valid`
  Especifica el modo de procesamiento XML que debe utilizarse para el documento de instancia XML: `wf`=comprobación de formato; `id`=comprobación de formato con ID/IDREF; `valid`=validación. Valor predeterminado: `wf`. Recuerde que el valor `valid` exige que cada documento de esquema que se cargue durante el procesamiento haga referencia a una DTD. Si no existe ninguna DTD, se generará un error.

- xml-validation-error-as-warning

  `--xml-validation-error-as-warning =` `true|false`
  Si es `true`, tratar los errores de validación como advertencias. Si los errores se tratan como advertencias, el procesamiento adicional como la transformación XSLT, proseguirá independientemente de los errores. Por defecto es `false`.

- xsd

  `--xsd =` *ARCHIVO*
  Especifica qué esquemas XML deben utilizarse para la validación de documentos XML. Si quiere especificar más de un esquema, añada la opción varias veces.

- xsd-version

  `--xsd-version =` `1.0|1.1|detect`

  Especifica qué versión de la especificación Schema Definition Language (XSD) del W3C se debe usar. Valor predeterminado: `1.0`.

  Esta opción también puede ser útil si quiere ver en qué aspectos no es compatible un esquema 1.0 con la especificación 1.1. El valor `detect` es una característica de Altova. Permite detectar la versión

 © 2017-2023 Altova GmbH

del esquema XML (`1.0` o `1.1`) leyendo el valor del atributo `vc:minVersion` del elemento `<xs:schema>` del documento. Si el valor del atributo `@vc:minVersion` es `1.1`, se entiende que la versión del esquema es `1.1`. Si el atributo tiene otro valor que no sea `1.1` (o si no está presente el atributo `@vc:minVersion`), se entiende que la versión del esquema es `1.0`.

Catálogos y recursos globales

catalog

`--catalog =` *ARCHIVO*
Especifica la ruta de acceso absoluta a un archivo de catálogo que no está en el archivo de catálogo raíz instalado. El valor predeterminado es la ruta de acceso absoluta del archivo de catálogo raíz instalado (`<carpeta-instalación>\Altova\RaptorXMLServer2024\etc\RootCatalog.xml`). Consulte el apartado Catálogos XML [48] para obtener más información.

user-catalog

`--user-catalog =` *ARCHIVO*
Especifica la ruta de acceso absoluta a un catálogo XML que debe utilizarse junto con el catálogo raíz. Consulte el apartado Catálogos XML [48] para obtener más información.

enable-globalresources

`--enable-globalresources =` `true|false`
Habilita la función de recursos globales [55]. Valor predeterminado: `false`.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

globalresourceconfig [gc]

`--gc | --globalresourceconfig =` *VALOR*
Especifica la configuración activa del recurso global [55] (y habilita los recursos globales) [55].

globalresourcefile [gr]

`--gr | --globalresourcefile =` *ARCHIVO*
Especifica el archivo de recursos globales [55] (y habilita los recursos globales) [55].

Extensiones

Estas opciones definen cómo se gestionan las funciones de extensión especiales disponibles en la edición Enterprise Edition de varios productos de Altova (como XMLSpy Enterprise Edition). Su uso se describe detalladamente en el manual del usuario de cada producto.

chartext-disable

`--chartext-disable =` `true|false`
Deshabilita las extensiones de gráficos. Valor predeterminado: `false`.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

dotnetext-disable

--dotnetext-disable = true|false
Deshabilita las extensiones .NET. Valor predeterminado: false.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en true.

▼ jvm-location

--jvm-location = *ARCHIVO*
ARCHIVO especifica la ubicación del equipo virtual Java (*DLL* en Windows, objeto *compartido (shared object)* en Linux. El JVM se necesita si usa las funciones de extensión Java 512 en su código XSLT/XQuery. El valor predeterminado es false.

▼ javaext-barcode-location

--javaext-barcode-location = *ARCHIVO*
Especifica la ruta de acceso de la carpeta que contiene el archivo de extensión de código de barras AltovaBarcodeExtension.jar. La ruta de acceso debe darse en uno de estos formatos:

- Un URI de archivo (ejemplo: --javaext-barcode-location="file:///C:/Archivos de programa/Altova/RaptorXMLServer2024/etc/jar/")
- Una ruta de acceso Windows con caracteres de escape para las barras diagonales inversas (ejemplo: --javaext-barcode-location="C:\\Archivos de programa\\Altova\ \RaptorXMLServer2024\\etc\\jar\\")

▼ javaext-disable

--javaext-disable = true|false
Deshabilita las extensiones Java. Valor predeterminado: false.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en true.

▼ Opciones comunes

▼ error-format

--error-format = text|shortxml|longxml
Especifica el formato de la salida de error. Los valores posibles son formatos de texto, XML y XML detallado (longxml). Valor predeterminado: text.

▼ error-limit

--error-limit = *N | unlimited*
Especifica el límite de errores con un valor comprendido entre 1 y 9999 o unlimited (ilimitado). El valor predeterminado es 100. Cuando se alcanza el límite de error, se detiene la validación. Esta opción es muy práctica a la hora de limitar el uso del procesador durante la validación o transformación.

▼ info-limit

--info-limit = *N | unlimited*
Indica el límite del mensaje de información dentro del rango 1-65535 or unlimited. Si se alcanza el límite de información indicado, el procesamiento continúa pero ya no se informa de más mensajes.

 © 2017-2023 Altova GmbH

El valor predeterminado es `100`.

- help

  `--help`
  Muestra el texto de ayuda para el comando. Por ejemplo `valany --h`. (Otra opción es usar el comando `help` con un argumento. Por ejemplo: `help valany`).

- listfile

  `--listfile =` `true|false`
  Si el valor es `true`, el argumento *`ArchivoEntrada`* del comando se entiende como un archivo de texto que contiene un nombre de archivo por línea. Otra opción es enumerar los archivos en la ILC, separados por un espacio. No obstante, recuerde que las ILC tienen un límite de caracteres. Además, no olvide que la opción `--listfile` solamente afecta a los argumentos y no a las opciones.
  Valor predeterminado: `false`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- log-output

  `--log-output =` *`ARCHIVO`*
  Escribe el registro de salida en la URL de archivo indicada. Compruebe que la ILC tiene permiso de escritura en la ubicación de destino.

- network-timeout

  `--network-timeout =` *`VALOR`*
  Indica el tiempo de espera en milisegundos para operaciones remotas de entrada y salida. Valor predeterminado: `40000`.

- recurse

  `--recurse =` `true|false`
  Esta opción se utiliza para seleccionar ficheros dentro de subdirectorios (incluso en archivos ZIP). Si el valor es `true`, el argumento *`ArchivoEntrada`* del comando seleccionará el fichero seleccionado también en los subdirectorios. Por ejemplo: `"test.zip|zip\test.xml"` seleccionará los ficheros llamados `test.xml` en todos los subdirectorios de la carpeta ZIP. Si quiere puede usar los caracteres comodín `*` y `?`. Por ejemplo: `*.xml` seleccionaría todos los ficheros de la carpeta ZIP que tengan la extensión `.xml`. Valor predeterminado: `false`
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- verbose

  `--verbose =` `true|false`
  Si el valor es `true`, se genera información adicional durante la validación. Valor predeterminado es `false`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- verbose-output

--verbose-output = *ARCHIVO*
Escribe el resultado detallado en el *ARCHIVO* indicado.

- version

  --version
  Muestra el número de versión de RaptorXML Server. Si se utiliza con un comando, escriba la opción `--version` antes del comando.

- warning-limit

  --warning-limit = *N | unlimited*
  Especifica el límite de advertencia en el rango `1-65535` o `unlimited` (ilimitado). El procesamiento continua si se alcanza el límite pero no se registrarán más advertencias. El valor por defecto es `100`.

5.3.3 valxquery

El comando **valxquery** toma un archivo XQuery como único argumento y lo valida.

Windows	**RaptorXML valxquery [opciones]** ***Archivo-XQuery***
Linux	**raptorxml valxquery [opciones]** ***Archivo-XQuery***
Mac	**raptorxml valxquery [opciones]** ***Archivo-XQuery***

El argumento *Archivo-XQuery* es la ruta de acceso y el nombre del archivo XQuery que debe validarse.

Ejemplos

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

- Mayúsculas/minúsculas y barras en la línea de comandos

  RaptorXML (y **RaptorXMLServer** para comandos administrativos) *en Windows*
  raptorxml (y **raptorxmlserver** para comandos administrativos) *en Windows y Unix (Linux, Mac)*

  * Las minúsculas (`raptorxml` y `raptorxmlserver`) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que las mayúsculas (`RaptorXML`) funcionan solamente en Windows y Mac.
  * Use la barra diagonal en Linux y Mac y la barra diagonal inversa en Windows.

- Barra diagonal inversa y espacios en sistemas Windows

  En sistemas Windows: si hay espacios o caracteres especiales (por ejemplo en los nombres de archivos o de carpetas, empresas, personas o productos) debe usar comillas: por ejemplo, **"Mi archivo"**. Sin embargo, debe tener en cuenta que una barra diagonal inversa seguida de comillas dobles (por ejemplo,

 © 2017-2023 Altova GmbH

`"C:\Mi Directorio\"`) es posible que no se lea correctamente. Esto se debe a que la barra diagonal inversa también se usa para empezar secuencias de escape y la secuencia de escape `\"` también usa las comillas dobles. Para evitar secuencia de caracteres puede añadir otra barra diagonal inversa: `\\"`. En resumen: si necesita escribir una ruta que contenga espacios y una barra diagonal inversa, esta es la mejor forma de hacerlo: `"C:\Mi Directorio\\"`.

Opciones

Las opciones del comando aparecen más abajo divididas en grupos. Los valores se pueden dar sin comillas excepto en estos dos casos: (i) cuando la cadena de valor contiene espacios y (ii) cuando en la descripción de la opción se indique explícitamente que es necesario el uso de comillas. Si una opción toma un valor booleano y no se indica ningún valor, entonces el valor predeterminado de la opción es `TRUE`. Use la opción **`--h, --help`** para ver información sobre el comando.

Procesamiento XQuery

omit-xml-declaration

`--omit-xml-declaration =` `true|false`
Opción de serialización que especifica si la declaración XML se omite en el resultado o no. Si el valor es `true`, el documento de salida no tendrá una declaración XML. Si el valor es `false`, se incluye una declaración XML en el documento de salida. Valor predeterminado: `false`.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

xquery-version

`--xquery-version =` `1|1.0|3|3.0|3.1`
Indica si el procesador XQuery debe usar XQuery 1.0 o 3.0. Valor predeterminado: `3.1`

Instancias y esquemas XML

load-xml-with-psvi

`--load-xml-with-psvi =` `true|false`
Habilita la validación de archivos XML de entrada y genera información posterior a la validación. Valor predeterminado: `true`.

schema-imports

`--schema-imports =` `load-by-schemalocation | load-preferring-schemalocation | load-by-namespace | load-combining-both | license-namespace-only`
Esta opción indica el comportamiento de los elementos `xs:import`. Cada uno de estos elementos tiene un atributo opcional `namespace` y un atributo opcional `schemaLocation`: `<import namespace="unEspacioNombres" schemaLocation="unaURL">`. La opción indica si se debe cargar un documento de esquema o solo autorizar a un espacio de nombres. Si la opción indica que se debe cargar un documento de esquema, entonces indica también qué información debe utilizarse para encontrar el documento de esquema. Valor predeterminado: `load-preferring-schemalocation`.

- `load-by-schemalocation`: el valor del atributo `schemaLocation` se utiliza para buscar el

esquema, teniendo en cuenta las asignaciones de catálogo[48]. Si está presente el atributo `namespace`, se importa el espacio de nombres (con licencia).
- `load-preferring-schemalocation`: si está presente, se utiliza el atributo `schemaLocation` teniendo en cuenta las asignaciones de catálogo[48]. Si no está presente el atributo `schemaLocation`, entonces se usa el valor del atributo `namespace` a través de las asignaciones de catálogo[48]. Este es el **valor predeterminado**.
- `load-by-namespace`: el valor del atributo `namespace` se utiliza para buscar el esquema por medio de una asignación de catálogo[48].
- `load-combining-both`: si el atributo `namespace` o `schemaLocation` tiene una asignación de catálogo[48], entonces se usa la asignación. Si ambos atributos tienen asignaciones de catálogo[48], entonces es el valor de la opción `--schema-mapping` (opción XML/XSD[240]) decide qué asignación se utiliza. Si no hay ninguna asignación de catálogo[48], entonces se usa el atributo `schemaLocation`.
- `license-namespace-only`: se importa el espacio de nombres. No se importa el documento de esquema.

▼ schemalocation-hints

`--schemalocation-hints =` `load-by-schemalocation | load-by-namespace | load-combining-both | ignore`
Determina el comportamiento predeterminado de los atributos `xsi:schemaLocation` y `xsi:noNamespaceSchemaLocation`. Indica si se debe cargar un documento de esquema y, si así es, indica qué información debe utilizarse para encontrarlo. Valor predeterminado: `load-by-schemalocation`.

- **Valor predeterminado**: `load-by-schemalocation`. Este valor toma la URL de la ubicación del esquema[403] de los atributos `xsi:schemaLocation` y `xsi:noNamespaceSchemaLocation` de los documentos de instancia XML.

- El valor `load-by-namespace` toma la parte de espacio de nombres[403] del atributo `xsi:schemaLocation` y una cadena vacía en el caso del atributo `xsi:noNamespaceSchemaLocation` y encuentra el esquema por medio de una asignación de catálogo[48].

- Si usa el valor `load-combining-both` y el espacio de nombres o la URL tienen una asignación de catálogo[48], se usa dicha asignación. Si ambos tienen asignaciones de catálogo[48], el valor de la opción `schema-mapping` (opción XML/XSD[240]) decide qué asignación se utiliza. Si ni el espacio de nombres ni la URL tiene una asignación de catálogo, se usa la URL.

- El valor `ignore` ignora los atributos `xsi:schemaLocation` y `xsi:noNamespaceSchemaLocation`.

▼ schema-mapping

`--schema-mapping =` `prefer-schemalocation | prefer-namespace`
Si se usa la ubicación y el espacio de nombres para buscar el documento de esquema, esta opción indica cuál de ellos debe ser la opción preferida durante la búsqueda en el catálogo.
Si la opción `--schemalocation-hints` o la opción `--schema-imports` tiene el valor `load-combining-both` y si las partes de espacio de nombres y URL pertinentes tienen asignaciones de catálogo[48], entonces el valor de la opción especifica cuál de las dos asignaciones se utiliza (la asignación del espacio de nombres o de la URL: el valor `prefer-schemalocation` se refiere a la asignación de la URL). Valor predeterminado: `prefer-schemalocation`.

▼ xinclude

 © 2017-2023 Altova GmbH

--xinclude = true|false
Habilita la compatibilidad con inclusiones XML (XInclude). Si el valor es false, los elementos XInclude include se ignoran. Valor predeterminado: false.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en true.

- xml-mode

  --xml-mode = wf|id|valid
  Especifica el modo de procesamiento XML que debe utilizarse para el documento de instancia XML: wf=comprobación de formato; id=comprobación de formato con ID/IDREF; valid=validación. Valor predeterminado: wf. Recuerde que el valor valid exige que cada documento de instancia que se cargue durante el procesamiento haga referencia a una DTD. Si no existe ninguna DTD, se generará un error.

- xml-mode-for-schemas

  --xml-mode-for-schemas = wf|id|valid
  Especifica el modo de procesamiento XML que debe utilizarse para el documento de instancia XML: wf=comprobación de formato; id=comprobación de formato con ID/IDREF; valid=validación. Valor predeterminado: wf. Recuerde que el valor valid exige que cada documento de esquema que se cargue durante el procesamiento haga referencia a una DTD. Si no existe ninguna DTD, se generará un error.

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

  --xpath-static-type-errors-as-warnings = true|falsetrue|false
  Si se establece en true reduce a advertencia cualquier error que se detecte en el contexto estático XPath. Mientras que un error provocaría un error, una advertencia permite que el procesamiento continúe. El valor predeterminado es false.

- xsd-version

  --xsd-version = 1.0|1.1|detect

  Especifica qué versión de la especificación Schema Definition Language (XSD) del W3C se debe usar. Valor predeterminado: 1.0.

  Esta opción también puede ser útil si quiere ver en qué aspectos no es compatible un esquema 1.0 con la especificación 1.1. El valor detect es una característica de Altova. Permite detectar la versión del esquema XML (1.0 o 1.1) leyendo el valor del atributo vc:minVersion del elemento <xs:schema> del documento. Si el valor del atributo @vc:minVersion es 1.1, se entiende que la versión del esquema es 1.1. Si el atributo tiene otro valor que no sea 1.1 (o si no está presente el atributo @vc:minVersion), se entiende que la versión del esquema es 1.0.

- Catálogos y recursos globales
  - catalog

    --catalog = *ARCHIVO*
    Especifica la ruta de acceso absoluta a un archivo de catálogo que no está en el archivo de catálogo raíz instalado. El valor predeterminado es la ruta de acceso absoluta del archivo de catálogo raíz instalado (<carpeta-instalación>\Altova\RaptorXMLServer2024\etc\RootCatalog.xml).

Consulte el apartado Catálogos XML [48] para obtener más información.

- user-catalog

  --user-catalog = *ARCHIVO*
  Especifica la ruta de acceso absoluta a un catálogo XML que debe utilizarse junto con el catálogo raíz. Consulte el apartado Catálogos XML [48] para obtener más información.

- enable-globalresources

  --enable-globalresources = `true|false`
  Habilita la función de recursos globales [55]. Valor predeterminado: `false`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- globalresourceconfig [gc]

  --gc | --globalresourceconfig = *VALOR*
  Especifica la configuración activa del recurso global [55] (y habilita los recursos globales) [55].

- globalresourcefile [gr]

  --gr | --globalresourcefile = *ARCHIVO*
  Especifica el archivo de recursos globales [55] (y habilita los recursos globales) [55].

- Extensiones

  Estas opciones definen cómo se gestionan las funciones de extensión especiales disponibles en la edición Enterprise Edition de varios productos de Altova (como XMLSpy Enterprise Edition). Su uso se describe detalladamente en el manual del usuario de cada producto.

  - chartext-disable

    --chartext-disable = `true|false`
    Deshabilita las extensiones de gráficos. Valor predeterminado: `false`.
    Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

  - dotnetext-disable

    --dotnetext-disable = `true|false`
    Deshabilita las extensiones .NET. Valor predeterminado: `false`.
    Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

  - jvm-location

    --jvm-location = *ARCHIVO*
    ARCHIVO especifica la ubicación del equipo virtual Java (*DLL* en Windows, objeto *compartido (shared object)* en Linux. El JVM se necesita si usa las funciones de extensión Java [512] en su código XSLT/XQuery. El valor predeterminado es `false`.

  - javaext-barcode-location

 © 2017-2023 Altova GmbH

--javaext-barcode-location = *ARCHIVO*
Especifica la ruta de acceso de la carpeta que contiene el archivo de extensión de código de barras `AltovaBarcodeExtension.jar`. La ruta de acceso debe darse en uno de estos formatos:

- Un URI de archivo (ejemplo: `--javaext-barcode-location="file:///C:/Archivos de programa/Altova/RaptorXMLServer2024/etc/jar/"`)
- Una ruta de acceso Windows con caracteres de escape para las barras diagonales inversas (ejemplo: `--javaext-barcode-location="C:\\Archivos de programa\\Altova\ \RaptorXMLServer2024\\etc\\jar\\"`)

▼ javaext-disable

--javaext-disable = `true|false`
Deshabilita las extensiones Java. Valor predeterminado: `false`.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

▼ Opciones comunes

▼ error-format

--error-format = `text|shortxml|longxml`
Especifica el formato de la salida de error. Los valores posibles son formatos de texto, XML y XML detallado (`longxml`). Valor predeterminado: `text`.

▼ error-limit

--error-limit = *N | unlimited*
Especifica el límite de errores con un valor comprendido entre `1` y `9999` o `unlimited` (ilimitado). El valor predeterminado es `100`. Cuando se alcanza el límite de error, se detiene la validación. Esta opción es muy práctica a la hora de limitar el uso del procesador durante la validación o transformación.

▼ info-limit

--info-limit = *N | unlimited*
Indica el límite del mensaje de información dentro del rango `1-65535` or `unlimited`. Si se alcanza el límite de información indicado, el procesamiento continúa pero ya no se informa de más mensajes. El valor predeterminado es `100`.

▼ help

--help
Muestra el texto de ayuda para el comando. Por ejemplo `valany --h`. (Otra opción es usar el comando `help` con un argumento. Por ejemplo: `help valany`).

▼ listfile

--listfile = `true|false`
Si el valor es `true`, el argumento *ArchivoEntrada* del comando se entiende como un archivo de texto que contiene un nombre de archivo por línea. Otra opción es enumerar los archivos en la ILC, separados por un espacio. No obstante, recuerde que las ILC tienen un límite de caracteres.

Además, no olvide que la opción `--listfile` solamente afecta a los argumentos y no a las opciones.
Valor predeterminado: `false`.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- log-output

  `--log-output =` *`ARCHIVO`*
  Escribe el registro de salida en la URL de archivo indicada. Compruebe que la ILC tiene permiso de escritura en la ubicación de destino.

- network-timeout

  `--network-timeout =` *`VALOR`*
  Indica el tiempo de espera en milisegundos para operaciones remotas de entrada y salida. Valor predeterminado: `40000`.

- recurse

  `--recurse =` `true|false`
  Esta opción se utiliza para seleccionar ficheros dentro de subdirectorios (incluso en archivos ZIP). Si el valor es `true`, el argumento *`ArchivoEntrada`* del comando seleccionará el fichero seleccionado también en los subdirectorios. Por ejemplo: `"test.zip|zip\test.xml"` seleccionará los ficheros llamados `test.xml` en todos los subdirectorios de la carpeta ZIP. Si quiere puede usar los caracteres comodín `*` y `?`. Por ejemplo: `*.xml` seleccionaría todos los ficheros de la carpeta ZIP que tengan la extensión `.xml`. Valor predeterminado: `false`
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- verbose

  `--verbose =` `true|false`
  Si el valor es `true`, se genera información adicional durante la validación. Valor predeterminado es `false`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- verbose-output

  `--verbose-output =` *`ARCHIVO`*
  Escribe el resultado detallado en el *`ARCHIVO`* indicado.

- version

  `--version`
  Muestra el número de versión de RaptorXML Server. Si se utiliza con un comando, escriba la opción `--version` antes del comando.

- warning-limit

  `--warning-limit =` *`N | unlimited`*
  Especifica el límite de advertencia en el rango `1-65535` o `unlimited` (ilimitado). El procesamiento

 © 2017-2023 Altova GmbH

continua si se alcanza el límite pero no se registrarán más advertencias. El valor por defecto es `100`.

5.3.4 valxqueryupdate

El comando **`valxqueryupdate`** toma un archivo XQuery como único argumento y lo valida.

Windows	**`RaptorXML valxqueryupdate [opciones] Archivo-XQuery`**
Linux	**`raptorxml valxqueryupdate [opciones] Archivo-XQuery`**
Mac	**`raptorxml valxqueryupdate [opciones] Archivo-XQuery`**

El argumento ***`Archivo-XQuery`*** es la ruta de acceso y el nombre del archivo XQuery que se debe validar.

Ejemplos

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

▼ Mayúsculas/minúsculas y barras en la línea de comandos

`RaptorXML` (y **`RaptorXMLServer`** para comandos administrativos) *en Windows*
`raptorxml` (y **`raptorxmlserver`** para comandos administrativos) *en Windows y Unix (Linux, Mac)*

* Las minúsculas (`raptorxml` y `raptorxmlserver`) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que las mayúsculas (`RaptorXML`) funcionan solamente en Windows y Mac.
* Use la barra diagonal en Linux y Mac y la barra diagonal inversa en Windows.

▼ Barra diagonal inversa y espacios en sistemas Windows

En sistemas Windows: si hay espacios o caracteres especiales (por ejemplo en los nombres de archivos o de carpetas, empresas, personas o productos) debe usar comillas: por ejemplo, **`"Mi archivo"`**. Sin embargo, debe tener en cuenta que una barra diagonal inversa seguida de comillas dobles (por ejemplo, **`"C:\Mi Directorio\"`**) es posible que no se lea correctamente. Esto se debe a que la barra diagonal inversa también se usa para empezar secuencias de escape y la secuencia de escape **`\"`** también usa las comillas dobles. Para evitar secuencia de caracteres puede añadir otra barra diagonal inversa: **`\\"`**. En resumen: si necesita escribir una ruta que contenga espacios y una barra diagonal inversa, esta es la mejor forma de hacerlo: **`"C:\Mi Directorio\\"`**.

Opciones

Las opciones del comando aparecen más abajo divididas en grupos. Los valores se pueden dar sin comillas excepto en estos dos casos: (i) cuando la cadena de valor contiene espacios y (ii) cuando en la descripción de la opción se indique explícitamente que es necesario el uso de comillas. Si una opción toma un valor booleano

y no se indica ningún valor, entonces el valor predeterminado de la opción es `TRUE`. Use la opción **`--h, --help`** para ver información sobre el comando.

- Procesamiento XQuery
  - omit-xml-declaration

    `--omit-xml-declaration =` `true|false`
    Opción de serialización que especifica si la declaración XML se omite en el resultado o no. Si el valor es `true`, el documento de salida no tendrá una declaración XML. Si el valor es `false`, se incluye una declaración XML en el documento de salida. Valor predeterminado: `false`.
    Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

  - xquery-update-version

    `--xquery-update-version =` `1|1.0|3|3.0|`
    Indica si el procesador XQuery debería usar XQuery Update Facility 1.0 o XQuery Update Facility 3.0. Valor predeterminado: `3`.

- Instancias y esquemas XML
  - load-xml-with-psvi

    `--load-xml-with-psvi =` `true|false`
    Habilita la validación de archivos XML de entrada y genera información posterior a la validación. Valor predeterminado: `true`.

  - schema-imports

    `--schema-imports =` `load-by-schemalocation | load-preferring-schemalocation | load-by-namespace | load-combining-both | license-namespace-only`
    Esta opción indica el comportamiento de los elementos `xs:import`. Cada uno de estos elementos tiene un atributo opcional `namespace` y un atributo opcional `schemaLocation`: `<import namespace="unEspacioNombres" schemaLocation="unaURL">`. La opción indica si se debe cargar un documento de esquema o solo autorizar a un espacio de nombres. Si la opción indica que se debe cargar un documento de esquema, entonces indica también qué información debe utilizarse para encontrar el documento de esquema. Valor predeterminado: `load-preferring-schemalocation`.

    - `load-by-schemalocation`: el valor del atributo `schemaLocation` se utiliza para buscar el esquema, teniendo en cuenta las asignaciones de catálogo[48]. Si está presente el atributo `namespace`, se importa el espacio de nombres (con licencia).
    - `load-preferring-schemalocation`: si está presente, se utiliza el atributo `schemaLocation` teniendo en cuenta las asignaciones de catálogo[48]. Si no está presente el atributo `schemaLocation`, entonces se usa el valor del atributo `namespace` a través de las asignaciones de catálogo[48]. Este es el **valor predeterminado**.
    - `load-by-namespace`: el valor del atributo `namespace` se utiliza para buscar el esquema por medio de una asignación de catálogo[48].
    - `load-combining-both`: si el atributo `namespace` o `schemaLocation` tiene una asignación de catálogo[48], entonces se usa la asignación. Si ambos atributos tienen asignaciones de catálogo[48], entonces es el valor de la opción `--schema-mapping` (opción XML/XSD[240]) decide qué asignación se utiliza. Si no hay ninguna asignación de catálogo[48], entonces se usa el

 © 2017-2023 Altova GmbH

atributo `schemaLocation`.
- `license-namespace-only`: se importa el espacio de nombres. No se importa el documento de esquema.

▼ schemalocation-hints

`--schemalocation-hints =` `load-by-schemalocation | load-by-namespace | load-combining-both | ignore`
Determina el comportamiento predeterminado de los atributos `xsi:schemaLocation` y `xsi:noNamespaceSchemaLocation`. Indica si se debe cargar un documento de esquema y, si así es, indica qué información debe utilizarse para encontrarlo. Valor predeterminado: `load-by-schemalocation`.

- **Valor predeterminado**: `load-by-schemalocation`. Este valor toma la URL de la ubicación del esquema[403] de los atributos `xsi:schemaLocation` y `xsi:noNamespaceSchemaLocation` de los documentos de instancia XML.

- El valor `load-by-namespace` toma la parte de espacio de nombres[403] del atributo `xsi:schemaLocation` y una cadena vacía en el caso del atributo `xsi:noNamespaceSchemaLocation` y encuentra el esquema por medio de una asignación de catálogo[48].

- Si usa el valor `load-combining-both` y el espacio de nombres o la URL tienen una asignación de catálogo[48], se usa dicha asignación. Si ambos tienen asignaciones de catálogo[48], el valor de la opción `schema-mapping` (opción XML/XSD[240]) decide qué asignación se utiliza. Si ni el espacio de nombres ni la URL tiene una asignación de catálogo, se usa la URL.

- El valor `ignore` ignora los atributos `xsi:schemaLocation` y `xsi:noNamespaceSchemaLocation`.

▼ schema-mapping

`--schema-mapping =` `prefer-schemalocation | prefer-namespace`
Si se usa la ubicación y el espacio de nombres para buscar el documento de esquema, esta opción indica cuál de ellos debe ser la opción preferida durante la búsqueda en el catálogo.
Si la opción `--schemalocation-hints` o la opción `--schema-imports` tiene el valor `load-combining-both` y si las partes de espacio de nombres y URL pertinentes tienen asignaciones de catálogo[48], entonces el valor de la opción especifica cuál de las dos asignaciones se utiliza (la asignación del espacio de nombres o de la URL: el valor `prefer-schemalocation` se refiere a la asignación de la URL). Valor predeterminado: `prefer-schemalocation`.

▼ xinclude

`--xinclude =` `true|false`
Habilita la compatibilidad con inclusiones XML (XInclude). Si el valor es `false`, los elementos XInclude `include` se ignoran. Valor predeterminado: `false`.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

▼ xml-mode

`--xml-mode =` `wf|id|valid`
Especifica el modo de procesamiento XML que debe utilizarse para el documento de instancia XML: `wf`=comprobación de formato; `id`=comprobación de formato con ID/IDREF; `valid`=validación. Valor predeterminado: `wf`. Recuerde que el valor `valid` exige que cada documento de instancia que se

cargue durante el procesamiento haga referencia a una DTD. Si no existe ninguna DTD, se generará un error.

- xml-mode-for-schemas

  --xml-mode-for-schemas = `wf|id|valid`
  Especifica el modo de procesamiento XML que debe utilizarse para el documento de instancia XML: `wf`=comprobación de formato; `id`=comprobación de formato con ID/IDREF; `valid`=validación. Valor predeterminado: `wf`. Recuerde que el valor `valid` exige que cada documento de esquema que se cargue durante el procesamiento haga referencia a una DTD. Si no existe ninguna DTD, se generará un error.

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

  --xpath-static-type-errors-as-warnings = `true|falsetrue|false`
  Si se establece en `true` reduce a advertencia cualquier error que se detecte en el contexto estático XPath. Mientras que un error provocaría un error, una advertencia permite que el procesamiento continúe. El valor predeterminado es `false`.

- xsd-version

  --xsd-version = `1.0|1.1|detect`

  Especifica qué versión de la especificación Schema Definition Language (XSD) del W3C se debe usar. Valor predeterminado: `1.0`.

  Esta opción también puede ser útil si quiere ver en qué aspectos no es compatible un esquema 1.0 con la especificación 1.1. El valor `detect` es una característica de Altova. Permite detectar la versión del esquema XML (`1.0` o `1.1`) leyendo el valor del atributo `vc:minVersion` del elemento `<xs:schema>` del documento. Si el valor del atributo `@vc:minVersion` es `1.1`, se entiende que la versión del esquema es `1.1`. Si el atributo tiene otro valor que no sea `1.1` (o si no está presente el atributo `@vc:minVersion`), se entiende que la versión del esquema es `1.0`.

- Catálogos y recursos globales

  - catalog

    --catalog = *ARCHIVO*
    Especifica la ruta de acceso absoluta a un archivo de catálogo que no está en el archivo de catálogo raíz instalado. El valor predeterminado es la ruta de acceso absoluta del archivo de catálogo raíz instalado (`<carpeta-instalación>\Altova\RaptorXMLServer2024\etc\RootCatalog.xml`). Consulte el apartado Catálogos XML 48 para obtener más información.

  - user-catalog

    --user-catalog = *ARCHIVO*
    Especifica la ruta de acceso absoluta a un catálogo XML que debe utilizarse junto con el catálogo raíz. Consulte el apartado Catálogos XML 48 para obtener más información.

  - enable-globalresources

    --enable-globalresources = `true|false`
    Habilita la función de recursos globales 55. Valor predeterminado: `false`.

 © 2017-2023 Altova GmbH

Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- globalresourceconfig [gc]

  `--gc | --globalresourceconfig =` *`VALOR`*
  Especifica la configuración activa del recurso global[55] (y habilita los recursos globales)[55].

- globalresourcefile [gr]

  `--gr | --globalresourcefile =` *`ARCHIVO`*
  Especifica el archivo de recursos globales[55] (y habilita los recursos globales)[55].

- Extensiones

  Estas opciones definen cómo se gestionan las funciones de extensión especiales disponibles en la edición Enterprise Edition de varios productos de Altova (como XMLSpy Enterprise Edition). Su uso se describe detalladamente en el manual del usuario de cada producto.

  - chartext-disable

    `--chartext-disable =` `true|false`
    Deshabilita las extensiones de gráficos. Valor predeterminado: `false`.
    Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

  - dotnetext-disable

    `--dotnetext-disable =` `true|false`
    Deshabilita las extensiones .NET. Valor predeterminado: `false`.
    Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

  - jvm-location

    `--jvm-location =` *`ARCHIVO`*
    `ARCHIVO` especifica la ubicación del equipo virtual Java (*DLL* en Windows, objeto *compartido (shared object)* en Linux. El JVM se necesita si usa las funciones de extensión Java[512] en su código XSLT/XQuery. El valor predeterminado es `false`.

  - javaext-barcode-location

    `--javaext-barcode-location =` *`ARCHIVO`*
    Especifica la ruta de acceso de la carpeta que contiene el archivo de extensión de código de barras `AltovaBarcodeExtension.jar`. La ruta de acceso debe darse en uno de estos formatos:

    - Un URI de archivo (ejemplo: `--javaext-barcode-location="file:///C:/Archivos de programa/Altova/RaptorXMLServer2024/etc/jar/"`)
    - Una ruta de acceso Windows con caracteres de escape para las barras diagonales inversas (ejemplo: `--javaext-barcode-location="C:\\Archivos de programa\\Altova\ \RaptorXMLServer2024\\etc\\jar\\"`)

  - javaext-disable

--javaext-disable = `true|false`
Deshabilita las extensiones Java. Valor predeterminado: `false`.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- Opciones comunes
  - error-format

    --error-format = `text|shortxml|longxml`
    Especifica el formato de la salida de error. Los valores posibles son formatos de texto, XML y XML detallado (`longxml`). Valor predeterminado: `text`.

  - error-limit

    --error-limit = *N | unlimited*
    Especifica el límite de errores con un valor comprendido entre `1` y `9999` o `unlimited` (ilimitado). El valor predeterminado es `100`. Cuando se alcanza el límite de error, se detiene la validación. Esta opción es muy práctica a la hora de limitar el uso del procesador durante la validación o transformación.

  - info-limit

    --info-limit = *N | unlimited*
    Indica el límite del mensaje de información dentro del rango `1-65535` or `unlimited`. Si se alcanza el límite de información indicado, el procesamiento continúa pero ya no se informa de más mensajes. El valor predeterminado es `100`.

  - help

    --help
    Muestra el texto de ayuda para el comando. Por ejemplo `valany --h`. (Otra opción es usar el comando `help` con un argumento. Por ejemplo: `help valany`).

  - listfile

    --listfile = `true|false`
    Si el valor es `true`, el argumento *ArchivoEntrada* del comando se entiende como un archivo de texto que contiene un nombre de archivo por línea. Otra opción es enumerar los archivos en la ILC, separados por un espacio. No obstante, recuerde que las ILC tienen un límite de caracteres. Además, no olvide que la opción `--listfile` solamente afecta a los argumentos y no a las opciones.
    Valor predeterminado: `false`.
    Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

  - log-output

    --log-output = *ARCHIVO*
    Escribe el registro de salida en la URL de archivo indicada. Compruebe que la ILC tiene permiso de escritura en la ubicación de destino.

© 2017-2023 Altova GmbH

- network-timeout

  --network-timeout = *VALOR*
  Indica el tiempo de espera en milisegundos para operaciones remotas de entrada y salida. Valor predeterminado: `40000`.

- recurse

  --recurse = `true|false`
  Esta opción se utiliza para seleccionar ficheros dentro de subdirectorios (incluso en archivos ZIP). Si el valor es `true`, el argumento *ArchivoEntrada* del comando seleccionará el fichero seleccionado también en los subdirectorios. Por ejemplo: `"test.zip|zip\test.xml"` seleccionará los ficheros llamados `test.xml` en todos los subdirectorios de la carpeta ZIP. Si quiere puede usar los caracteres comodín `*` y `?`. Por ejemplo: `*.xml` seleccionaría todos los ficheros de la carpeta ZIP que tengan la extensión `.xml`. Valor predeterminado: `false`
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- verbose

  --verbose = `true|false`
  Si el valor es `true`, se genera información adicional durante la validación. Valor predeterminado es `false`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- verbose-output

  --verbose-output = *ARCHIVO*
  Escribe el resultado detallado en el *ARCHIVO* indicado.

- version

  --version
  Muestra el número de versión de RaptorXML Server. Si se utiliza con un comando, escriba la opción `--version` antes del comando.

- warning-limit

  --warning-limit = *N | unlimited*
  Especifica el límite de advertencia en el rango `1-65535` o `unlimited` (ilimitado). El procesamiento continua si se alcanza el límite pero no se registrarán más advertencias. El valor por defecto es `100`.

5.4 Comandos XSLT

Estos son los comandos XSLT:

xslt 128	Para transformar documentos XML con un documento XSLT.
valxslt 137	Para validar documentos XSLT.

Los argumentos y opciones de cada comando se describen en los siguientes apartados.

5.4.1 xslt

El comando **xslt** toma un archivo XSLT como único argumento y lo utiliza para transformar un archivo XML de entrada y generar un archivo de salida. Los archivos de entrada y salida se especifican como opciones 245.

Windows	**RaptorXML xslt [opciones]** ***Archivo-XSLT***
Linux	**raptorxml xslt [opciones]** ***Archivo-XSLT***
Mac	**raptorxml xslt [opciones]** ***Archivo-XSLT***

- El argumento Archivo-XSLT es la ruta de acceso y el nombre del archivo XSLT que se debe usar para la transformación.
- Se necesita un archivo XML de entrada (--input 245) o el punto de entrada de una plantilla con nombre (--template-entry-point 245).
- Para transformar datos JSON, cárguelos con la función **json-doc($path)** de XPath 3.1 y use la opción **--initial-match-selection** del comando **xslt**. Consulte el último elemento de los ejemplos siguientes.
- Si no especifica la opción del documento de salida --output 245, se genera un resultado estándar. Puede usar XSLT 1.0, 2.0 o 3.0. La versión predeterminada es XSLT 3.0.

Ejemplos

- **raptorxml** xslt --input=c:\Test.xml --output=c:\Salida.xml c:\Test.xslt
- **raptorxml** xslt --template-entry-point=PlantillaInicial --output=c:\Salida.xml c:\Test.xslt
- **raptorxml** xslt --input=c:\Test.xml --output=c:\Salida.xml --param=date://node[1]/@att1 --p=title:'cadenasinespacios' --param=title:"'cadena con espacios'" --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 (Si la cadena del argumento `json-doc` contiene spacios, las comillas deben abarcar todo el valor `json-doc`.)

▼ Mayúsculas/minúsculas y barras en la línea de comandos

 © 2017-2023 Altova GmbH

RaptorXML (y **RaptorXMLServer** para comandos administrativos) *en Windows*
raptorxml (y **raptorxmlserver** para comandos administrativos) *en Windows y Unix (Linux, Mac)*

* Las minúsculas (raptorxml y raptorxmlserver) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que las mayúsculas (RaptorXML) funcionan solamente en Windows y Mac.
* Use la barra diagonal en Linux y Mac y la barra diagonal inversa en Windows.

▼ Barra diagonal inversa y espacios en sistemas Windows

En sistemas Windows: si hay espacios o caracteres especiales (por ejemplo en los nombres de archivos o de carpetas, empresas, personas o productos) debe usar comillas: por ejemplo, **"Mi archivo"**. Sin embargo, debe tener en cuenta que una barra diagonal inversa seguida de comillas dobles (por ejemplo, **"C:\Mi Directorio\"**) es posible que no se lea correctamente. Esto se debe a que la barra diagonal inversa también se usa para empezar secuencias de escape y la secuencia de escape **\"** también usa las comillas dobles. Para evitar secuencia de caracteres puede añadir otra barra diagonal inversa: **\\"**. En resumen: si necesita escribir una ruta que contenga espacios y una barra diagonal inversa, esta es la mejor forma de hacerlo: **"C:\Mi Directorio\\"**.

Opciones

Las opciones del comando aparecen más abajo divididas en grupos. Los valores se pueden dar sin comillas excepto en estos dos casos: (i) cuando la cadena de valor contiene espacios y (ii) cuando en la descripción de la opción se indique explícitamente que es necesario el uso de comillas. Si una opción toma un valor booleano y no se indica ningún valor, entonces el valor predeterminado de la opción es TRUE. Use la opción **--h, --help** para ver información sobre el comando.

▼ Procesamiento XSLT

▼ indent-characters

--indent-characters = *VALOR*
Especifica la cadena de caracteres que debe usarse como sangría.

▼ function-param

--function-param = *VALUE*
Especifica las funciones que se transmiten a la función inicial. Para especificar más de una función, use la opción varias veces, pero recuerde que el orden es importante.

▼ global-context-item

--global-context-item = *VALUE*
Especifica el componente de contexto que se debe usar para evaluar las variables globales.

▼ initial-function

--initial-function = *VALUE*
El nombre de una función que se ejecuta como punto de entrada de la transformación.

▼ initial-match-selection

--initial-match-selection = *VALUE*

Especifica el valor (secuencia) de la selección de coincidencia inicial.

- initial-mode, template-mode

  --initial-mode, --template-mode = *VALOR*
  Especifica el modo de plantilla que debe usarse para la transformación.

- initial-template, template-entry-point

  --initial-template, --template-entry-point = *VALOR*
  Indica el nombre de una plantilla con nombre de la hoja de estilos XSLT que sirve de punto de entrada de la transformación.

- input

  --input = *ARCHIVO*
  La URL del archivo XML que se debe transformar.

- output, xsltoutput

  output = *ARCHIVO*, **xsltoutput =** *ARCHIVO*
  La URL del archivo de salida principal. Por ejemplo, en caso de tener varios archivos HTML de salida, el archivo de salida principal será la ubicación del archivo HTML del punto de entrada. Los demás archivos de salida (como archivos de imagen generados) se indican con `xslt-additional-output-files`. Si no se especifica la opción `--output` ni la opción `--xsloutput`, se genera un resultado estándar.

- param [p]

  --p | --param = *CLAVE:VALOR*

  - ***XQuery***

    Especifica el valor de un parámetro externo. En el documento XQuery los parámetros externos se declaran con la declaración `declare variable` seguida de un nombre de variable y después la palabra clave `external` seguida del punto y coma final. Por ejemplo: `declare variable $foo as xs:string external;`

    Al usar la palabra clave `external`, `$foo` se convierte en parámetro externo y su valor se pasa en tiempo de ejecución desde una fuente externa. El parámetro externo recibe un valor con el comando de la ILC. Por ejemplo: `--param=foo:'MiNombre'`

    En la descripción anterior, *CLAVE* es el nombre de parámetro externo y *VALOR* es su valor, dado como expresión XPath. Los nombres de parámetro utilizados en la ILC deben declararse en el documento XQuery. Si se pasan valores a varios parámetros externos en la ILC, cada parámetro debe llevar una opción `--param` distinta. Si la expresión XPath contiene espacios, entonces debe estar entre comillas dobles.

  - ***XSLT***

    Especifica un parámetro global de la hoja de estilos. *CLAVE* es el nombre del parámetro y *VALOR* es una expresión XPath que da un valor al parámetro. Los nombres de parámetro utilizados en la ILC deben declararse en la hoja de estilos. Si usa más de un parámetro, debe usar el modificador `--param` antes de cada parámetro. Si la expresión XPath incluye espacios,

 © 2017-2023 Altova GmbH

entonces debe ir entre comillas dobles, tanto si el espacio está en la expresión propiamente dicha o en un literal de cadena de la expresión. Por ejemplo:

```
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`
  Habilita la serialización de secuencias de datos. Valor predeterminado: `true`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- template-param

  --template-param = *KEY:VALUE*
  Especifica los parámetros que pasan únicamente a la plantilla inicial (y no a llamadas descendientes de la plantilla). Para especificar varios parámetros, use esta opción una vez por cada uno de ellos.

- tunnel-param

  --tunnel-param = *KEY:VALUE*
  Especifica parámetros que pasan a la plantilla inicial y a llamadas descendientes de la plantilla. Para especificar varios parámetros, use esta opción una vez por cada uno de ellos.

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

  --xpath-static-type-errors-as-warnings = `true|falsetrue|false`
  Si se establece en `true` reduce a advertencia cualquier error que se detecte en el contexto estático XPath. Mientras que un error provocaría un error, una advertencia permite que el procesamiento continúe. El valor predeterminado es `false`.

- xslt-version

  --xslt-version = `1|1.0|2|2.0|3|3.0|3.1`
  Especifica si el procesador XSLT debe usar XSLT 1.0, XSLT 2.0 o XSLT 3.0. Valor predeterminado: `3`

- Instancias y esquemas XML
  - load-xml-with-psvi

    --load-xml-with-psvi = `true|false`
    Habilita la validación de archivos XML de entrada y genera información posterior a la validación. Valor predeterminado: `true`.

  - schema-imports

    --schema-imports = `load-by-schemalocation | load-preferring-schemalocation | load-by-namespace | load-combining-both | license-namespace-only`
    Esta opción indica el comportamiento de los elementos `xs:import`. Cada uno de estos elementos tiene un atributo opcional `namespace` y un atributo opcional `schemaLocation`: `<import`

`namespace="unEspacioNombres" schemaLocation="unaURL">`. La opción indica si se debe cargar un documento de esquema o solo autorizar a un espacio de nombres. Si la opción indica que se debe cargar un documento de esquema, entonces indica también qué información debe utilizarse para encontrar el documento de esquema. Valor predeterminado: `load-preferring-schemalocation`.

- `load-by-schemalocation`: el valor del atributo `schemaLocation` se utiliza para buscar el esquema, teniendo en cuenta las asignaciones de catálogo[48]. Si está presente el atributo `namespace`, se importa el espacio de nombres (con licencia).
- `load-preferring-schemalocation`: si está presente, se utiliza el atributo `schemaLocation` teniendo en cuenta las asignaciones de catálogo[48]. Si no está presente el atributo `schemaLocation`, entonces se usa el valor del atributo `namespace` a través de las asignaciones de catálogo[48]. Este es el **valor predeterminado**.
- `load-by-namespace`: el valor del atributo `namespace` se utiliza para buscar el esquema por medio de una asignación de catálogo[48].
- `load-combining-both`: si el atributo `namespace` o `schemaLocation` tiene una asignación de catálogo[48], entonces se usa la asignación. Si ambos atributos tienen asignaciones de catálogo[48], entonces es el valor de la opción `--schema-mapping` (opción XML/XSD[240]) decide qué asignación se utiliza. Si no hay ninguna asignación de catálogo[48], entonces se usa el atributo `schemaLocation`.
- `license-namespace-only`: se importa el espacio de nombres. No se importa el documento de esquema.

▼ schemalocation-hints

`--schemalocation-hints =` `load-by-schemalocation | load-by-namespace | load-combining-both | ignore`

Determina el comportamiento predeterminado de los atributos `xsi:schemaLocation` y `xsi:noNamespaceSchemaLocation`. Indica si se debe cargar un documento de esquema y, si así es, indica qué información debe utilizarse para encontrarlo. Valor predeterminado: `load-by-schemalocation`.

- **Valor predeterminado**: `load-by-schemalocation`. Este valor toma la URL de la ubicación del esquema[403] de los atributos `xsi:schemaLocation` y `xsi:noNamespaceSchemaLocation` de los documentos de instancia XML.
- El valor `load-by-namespace` toma la parte de espacio de nombres[403] del atributo `xsi:schemaLocation` y una cadena vacía en el caso del atributo `xsi:noNamespaceSchemaLocation` y encuentra el esquema por medio de una asignación de catálogo[48].
- Si usa el valor `load-combining-both` y el espacio de nombres o la URL tienen una asignación de catálogo[48], se usa dicha asignación. Si ambos tienen asignaciones de catálogo[48], el valor de la opción `schema-mapping` (opción XML/XSD[240]) decide qué asignación se utiliza. Si ni el espacio de nombres ni la URL tiene una asignación de catálogo, se usa la URL.
- El valor `ignore` ignora los atributos `xsi:schemaLocation` y `xsi:noNamespaceSchemaLocation`.

▼ schema-mapping

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

Si se usa la ubicación y el espacio de nombres para buscar el documento de esquema, esta opción indica cuál de ellos debe ser la opción preferida durante la búsqueda en el catálogo.

 © 2017-2023 Altova GmbH

Si la opción `--schemalocation-hints` o la opción `--schema-imports` tiene el valor `load-combining-both` y si las partes de espacio de nombres y URL pertinentes tienen asignaciones de catálogo[48], entonces el valor de la opción especifica cuál de las dos asignaciones se utiliza (la asignación del espacio de nombres o de la URL: el valor `prefer-schemalocation` se refiere a la asignación de la URL). Valor predeterminado: `prefer-schemalocation`.

▼ xinclude

`--xinclude =` `true|false`
Habilita la compatibilidad con inclusiones XML (XInclude). Si el valor es `false`, los elementos XInclude `include` se ignoran. Valor predeterminado: `false`.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

▼ xml-mode

`--xml-mode =` `wf|id|valid`
Especifica el modo de procesamiento XML que debe utilizarse para el documento de instancia XML: `wf`=comprobación de formato; `id`=comprobación de formato con ID/IDREF; `valid`=validación. Valor predeterminado: `wf`. Recuerde que el valor `valid` exige que cada documento de instancia que se cargue durante el procesamiento haga referencia a una DTD. Si no existe ninguna DTD, se generará un error.

▼ xml-validation-error-as-warning

`--xml-validation-error-as-warning =` `true|false`
Si es `true`, tratar los errores de validación como advertencias. Si los errores se tratan como advertencias, el procesamiento adicional como la transformación XSLT, proseguirá independientemente de los errores. Por defecto es `false`.

▼ xsd

`--xsd =` *`ARCHIVO`*
Especifica qué esquemas XML deben utilizarse para la validación de documentos XML. Si quiere especificar más de un esquema, añada la opción varias veces.

▼ xsd-version

`--xsd-version =` `1.0|1.1|detect`

Especifica qué versión de la especificación Schema Definition Language (XSD) del W3C se debe usar. Valor predeterminado: `1.0`.

Esta opción también puede ser útil si quiere ver en qué aspectos no es compatible un esquema 1.0 con la especificación 1.1. El valor `detect` es una característica de Altova. Permite detectar la versión del esquema XML (`1.0` o `1.1`) leyendo el valor del atributo `vc:minVersion` del elemento `<xs:schema>` del documento. Si el valor del atributo `@vc:minVersion` es `1.1`, se entiende que la versión del esquema es `1.1`. Si el atributo tiene otro valor que no sea `1.1` (o si no está presente el atributo `@vc:minVersion`), se entiende que la versión del esquema es `1.0`.

▼ Catálogos y recursos globales

- catalog

  --catalog = *ARCHIVO*
  Especifica la ruta de acceso absoluta a un archivo de catálogo que no está en el archivo de catálogo raíz instalado. El valor predeterminado es la ruta de acceso absoluta del archivo de catálogo raíz instalado (`<carpeta-instalación>\Altova\RaptorXMLServer2024\etc\RootCatalog.xml`). Consulte el apartado Catálogos XML [48] para obtener más información.

- user-catalog

  --user-catalog = *ARCHIVO*
  Especifica la ruta de acceso absoluta a un catálogo XML que debe utilizarse junto con el catálogo raíz. Consulte el apartado Catálogos XML [48] para obtener más información.

- enable-globalresources

  --enable-globalresources = `true|false`
  Habilita la función de recursos globales [55]. Valor predeterminado: `false`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- globalresourceconfig [gc]

  --gc | --globalresourceconfig = *VALOR*
  Especifica la configuración activa del recurso global [55] (y habilita los recursos globales) [55].

- globalresourcefile [gr]

  --gr | --globalresourcefile = *ARCHIVO*
  Especifica el archivo de recursos globales [55] (y habilita los recursos globales) [55].

- Extensiones

  Estas opciones definen cómo se gestionan las funciones de extensión especiales disponibles en la edición Enterprise Edition de varios productos de Altova (como XMLSpy Enterprise Edition). Su uso se describe detalladamente en el manual del usuario de cada producto.

  - chartext-disable

    --chartext-disable = `true|false`
    Deshabilita las extensiones de gráficos. Valor predeterminado: `false`.
    Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

  - dotnetext-disable

    --dotnetext-disable = `true|false`
    Deshabilita las extensiones .NET. Valor predeterminado: `false`.
    Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

  - jvm-location

    --jvm-location = *ARCHIVO*
    ARCHIVO especifica la ubicación del equipo virtual Java (*DLL* en Windows, objeto *compartido (shared*

 © 2017-2023 Altova GmbH

object) en Linux. El JVM se necesita si usa las funciones de extensión Java [512] en su código XSLT/XQuery. El valor predeterminado es `false`.

- javaext-barcode-location

  `--javaext-barcode-location =` *`ARCHIVO`*
  Especifica la ruta de acceso de la carpeta que contiene el archivo de extensión de código de barras `AltovaBarcodeExtension.jar`. La ruta de acceso debe darse en uno de estos formatos:

  - Un URI de archivo (ejemplo: `--javaext-barcode-location="file:///C:/Archivos de programa/Altova/RaptorXMLServer2024/etc/jar/"`)
  - Una ruta de acceso Windows con caracteres de escape para las barras diagonales inversas (ejemplo: `--javaext-barcode-location="C:\\Archivos de programa\\Altova\\RaptorXMLServer2024\\etc\\jar\\"`)

- javaext-disable

  `--javaext-disable =` `true|false`
  Deshabilita las extensiones Java. Valor predeterminado: `false`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- Opciones comunes
  - error-format

    `--error-format =` `text|shortxml|longxml`
    Especifica el formato de la salida de error. Los valores posibles son formatos de texto, XML y XML detallado (`longxml`). Valor predeterminado: `text`.

  - error-limit

    `--error-limit =` *`N | unlimited`*
    Especifica el límite de errores con un valor comprendido entre `1` y `9999` o `unlimited` (ilimitado). El valor predeterminado es `100`. Cuando se alcanza el límite de error, se detiene la validación. Esta opción es muy práctica a la hora de limitar el uso del procesador durante la validación o transformación.

  - info-limit

    `--info-limit =` *`N | unlimited`*
    Indica el límite del mensaje de información dentro del rango `1-65535` or `unlimited`. Si se alcanza el límite de información indicado, el procesamiento continúa pero ya no se informa de más mensajes. El valor predeterminado es `100`.

  - help

    `--help`
    Muestra el texto de ayuda para el comando. Por ejemplo `valany --h`. (Otra opción es usar el comando `help` con un argumento. Por ejemplo: `help valany`).

  - listfile

--listfile = true|false
Si el valor es true, el argumento *ArchivoEntrada* del comando se entiende como un archivo de texto que contiene un nombre de archivo por línea. Otra opción es enumerar los archivos en la ILC, separados por un espacio. No obstante, recuerde que las ILC tienen un límite de caracteres. Además, no olvide que la opción --listfile solamente afecta a los argumentos y no a las opciones.
Valor predeterminado: false.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en true.

- log-output

--log-output = *ARCHIVO*
Escribe el registro de salida en la URL de archivo indicada. Compruebe que la ILC tiene permiso de escritura en la ubicación de destino.

- network-timeout

--network-timeout = *VALOR*
Indica el tiempo de espera en milisegundos para operaciones remotas de entrada y salida. Valor predeterminado: 40000.

- recurse

--recurse = true|false
Esta opción se utiliza para seleccionar ficheros dentro de subdirectorios (incluso en archivos ZIP). Si el valor es true, el argumento *ArchivoEntrada* del comando seleccionará el fichero seleccionado también en los subdirectorios. Por ejemplo: "test.zip|zip\test.xml" seleccionará los ficheros llamados test.xml en todos los subdirectorios de la carpeta ZIP. Si quiere puede usar los caracteres comodín * y ?. Por ejemplo: *.xml seleccionaría todos los ficheros de la carpeta ZIP que tengan la extensión .xml. Valor predeterminado: false
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en true.

- verbose

--verbose = true|false
Si el valor es true, se genera información adicional durante la validación. Valor predeterminado es false.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en true.

- verbose-output

--verbose-output = *ARCHIVO*
Escribe el resultado detallado en el *ARCHIVO* indicado.

- version

--version
Muestra el número de versión de RaptorXML Server. Si se utiliza con un comando, escriba la opción --version antes del comando.

 © 2017-2023 Altova GmbH

- warning-limit

  --warning-limit = *N | unlimited*
  Especifica el límite de advertencia en el rango `1-65535` o `unlimited` (ilimitado). El procesamiento continua si se alcanza el límite pero no se registrarán más advertencias. El valor por defecto es `100`.

5.4.2 valxslt

El comando **valxslt** toma un archivo XSLT como único argumento y lo valida.

Windows	**RaptorXML valxslt [opciones]** ***Archivo-XSLT***
Linux	**raptorxml valxslt [opciones]** ***Archivo-XSLT***
Mac	**raptorxml valxslt [opciones]** ***Archivo-XSLT***

El argumento *Archivo-XSLT* es la ruta de acceso y el nombre del archivo XSLT que debe validarse. El archivo se valida con la especificación XSLT 1.0, 2.0 o 3.0. La versión predeterminada es XSLT 3.0.

Ejemplos

- **raptorxml** valxslt c:\Test.xslt
- **raptorxml** valxslt --xslt-version=2 c:\Test.xslt

- Mayúsculas/minúsculas y barras en la línea de comandos

  RaptorXML (y **RaptorXMLServer** para comandos administrativos) *en Windows*
  raptorxml (y **raptorxmlserver** para comandos administrativos) *en Windows y Unix (Linux, Mac)*

  * Las minúsculas (`raptorxml` y `raptorxmlserver`) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que las mayúsculas (`RaptorXML`) funcionan solamente en Windows y Mac.
  * Use la barra diagonal en Linux y Mac y la barra diagonal inversa en Windows.

- Barra diagonal inversa y espacios en sistemas Windows

  En sistemas Windows: si hay espacios o caracteres especiales (por ejemplo en los nombres de archivos o de carpetas, empresas, personas o productos) debe usar comillas: por ejemplo, **"Mi archivo"**. Sin embargo, debe tener en cuenta que una barra diagonal inversa seguida de comillas dobles (por ejemplo, **"C:\Mi Directorio\"**) es posible que no se lea correctamente. Esto se debe a que la barra diagonal inversa también se usa para empezar secuencias de escape y la secuencia de escape **\"** también usa las comillas dobles. Para evitar secuencia de caracteres puede añadir otra barra diagonal inversa: **\\"**. En resumen: si necesita escribir una ruta que contenga espacios y una barra diagonal inversa, esta es la mejor forma de hacerlo: **"C:\Mi Directorio\\"**.

Opciones

Las opciones del comando aparecen más abajo divididas en grupos. Los valores se pueden dar sin comillas excepto en estos dos casos: (i) cuando la cadena de valor contiene espacios y (ii) cuando en la descripción de la opción se indique explícitamente que es necesario el uso de comillas. Si una opción toma un valor booleano y no se indica ningún valor, entonces el valor predeterminado de la opción es `TRUE`. Use la opción **`--h, --help`** para ver información sobre el comando.

Procesamiento XSLT

initial-template, template-entry-point

`--initial-template, --template-entry-point =` *VALOR*
Indica el nombre de una plantilla con nombre de la hoja de estilos XSLT que sirve de punto de entrada de la transformación.

initial-mode, template-mode

`--initial-mode, --template-mode =` *VALOR*
Especifica el modo de plantilla que debe usarse para la transformación.

xslt-version

`--xslt-version =` `1|1.0|2|2.0|3|3.0|3.1`
Especifica si el procesador XSLT debe usar XSLT 1.0, XSLT 2.0 o XSLT 3.0. Valor predeterminado: `3`

Instancias y esquemas XML

load-xml-with-psvi

`--load-xml-with-psvi =` `true|false`
Habilita la validación de archivos XML de entrada y genera información posterior a la validación. Valor predeterminado: `true`.

schema-imports

`--schema-imports =` `load-by-schemalocation | load-preferring-schemalocation | load-by-namespace | load-combining-both | license-namespace-only`
Esta opción indica el comportamiento de los elementos `xs:import`. Cada uno de estos elementos tiene un atributo opcional `namespace` y un atributo opcional `schemaLocation`: `<import namespace="unEspacioNombres" schemaLocation="unaURL">`. La opción indica si se debe cargar un documento de esquema o solo autorizar a un espacio de nombres. Si la opción indica que se debe cargar un documento de esquema, entonces indica también qué información debe utilizarse para encontrar el documento de esquema. Valor predeterminado: `load-preferring-schemalocation`.

- `load-by-schemalocation`: el valor del atributo `schemaLocation` se utiliza para buscar el esquema, teniendo en cuenta las asignaciones de catálogo[48]. Si está presente el atributo `namespace`, se importa el espacio de nombres (con licencia).
- `load-preferring-schemalocation`: si está presente, se utiliza el atributo `schemaLocation` teniendo en cuenta las asignaciones de catálogo[48]. Si no está presente el atributo `schemaLocation`, entonces se usa el valor del atributo `namespace` a través de las asignaciones de catálogo[48]. Este es el **valor predeterminado**.

© 2017-2023 Altova GmbH

- `load-by-namespace`: el valor del atributo `namespace` se utiliza para buscar el esquema por medio de una asignación de catálogo[48].
- `load-combining-both`: si el atributo `namespace` o `schemaLocation` tiene una asignación de catálogo[48], entonces se usa la asignación. Si ambos atributos tienen asignaciones de catálogo[48], entonces es el valor de la opción `--schema-mapping` (opción XML/XSD[240]) decide qué asignación se utiliza. Si no hay ninguna asignación de catálogo[48], entonces se usa el atributo `schemaLocation`.
- `license-namespace-only`: se importa el espacio de nombres. No se importa el documento de esquema.

▼ schemalocation-hints

`--schemalocation-hints =` `load-by-schemalocation | load-by-namespace | load-combining-both | ignore`
Determina el comportamiento predeterminado de los atributos `xsi:schemaLocation` y `xsi:noNamespaceSchemaLocation`. Indica si se debe cargar un documento de esquema y, si así es, indica qué información debe utilizarse para encontrarlo. Valor predeterminado: `load-by-schemalocation`.

- **Valor predeterminado**: `load-by-schemalocation`. Este valor toma la URL de la ubicación del esquema[403] de los atributos `xsi:schemaLocation` y `xsi:noNamespaceSchemaLocation` de los documentos de instancia XML.

- El valor `load-by-namespace` toma la parte de espacio de nombres[403] del atributo `xsi:schemaLocation` y una cadena vacía en el caso del atributo `xsi:noNamespaceSchemaLocation` y encuentra el esquema por medio de una asignación de catálogo[48].

- Si usa el valor `load-combining-both` y el espacio de nombres o la URL tienen una asignación de catálogo[48], se usa dicha asignación. Si ambos tienen asignaciones de catálogo[48], el valor de la opción `schema-mapping` (opción XML/XSD[240]) decide qué asignación se utiliza. Si ni el espacio de nombres ni la URL tiene una asignación de catálogo, se usa la URL.

- El valor `ignore` ignora los atributos `xsi:schemaLocation` y `xsi:noNamespaceSchemaLocation`.

▼ schema-mapping

`--schema-mapping =` `prefer-schemalocation | prefer-namespace`
Si se usa la ubicación y el espacio de nombres para buscar el documento de esquema, esta opción indica cuál de ellos debe ser la opción preferida durante la búsqueda en el catálogo.
Si la opción `--schemalocation-hints` o la opción `--schema-imports` tiene el valor `load-combining-both` y si las partes de espacio de nombres y URL pertinentes tienen asignaciones de catálogo[48], entonces el valor de la opción especifica cuál de las dos asignaciones se utiliza (la asignación del espacio de nombres o de la URL: el valor `prefer-schemalocation` se refiere a la asignación de la URL). Valor predeterminado: `prefer-schemalocation`.

▼ xinclude

`--xinclude =` `true|false`
Habilita la compatibilidad con inclusiones XML (XInclude). Si el valor es `false`, los elementos XInclude `include` se ignoran. Valor predeterminado: `false`.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- xml-mode

  --xml-mode = wf|id|valid
  Especifica el modo de procesamiento XML que debe utilizarse para el documento de instancia XML: wf=comprobación de formato; id=comprobación de formato con ID/IDREF; valid=validación. Valor predeterminado: wf. Recuerde que el valor valid exige que cada documento de instancia que se cargue durante el procesamiento haga referencia a una DTD. Si no existe ninguna DTD, se generará un error.

- xml-mode-for-schemas

  --xml-mode-for-schemas = wf|id|valid
  Especifica el modo de procesamiento XML que debe utilizarse para el documento de instancia XML: wf=comprobación de formato; id=comprobación de formato con ID/IDREF; valid=validación. Valor predeterminado: wf. Recuerde que el valor valid exige que cada documento de esquema que se cargue durante el procesamiento haga referencia a una DTD. Si no existe ninguna DTD, se generará un error.

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

  --xpath-static-type-errors-as-warnings = true|falsetrue|false
  Si se establece en true reduce a advertencia cualquier error que se detecte en el contexto estático XPath. Mientras que un error provocaría un error, una advertencia permite que el procesamiento continúe. El valor predeterminado es false.

- xsd-version

  --xsd-version = 1.0|1.1|detect

  Especifica qué versión de la especificación Schema Definition Language (XSD) del W3C se debe usar. Valor predeterminado: 1.0.

  Esta opción también puede ser útil si quiere ver en qué aspectos no es compatible un esquema 1.0 con la especificación 1.1. El valor detect es una característica de Altova. Permite detectar la versión del esquema XML (1.0 o 1.1) leyendo el valor del atributo vc:minVersion del elemento <xs:schema> del documento. Si el valor del atributo @vc:minVersion es 1.1, se entiende que la versión del esquema es 1.1. Si el atributo tiene otro valor que no sea 1.1 (o si no está presente el atributo @vc:minVersion), se entiende que la versión del esquema es 1.0.

- Catálogos y recursos globales
  - catalog

    --catalog = *ARCHIVO*
    Especifica la ruta de acceso absoluta a un archivo de catálogo que no está en el archivo de catálogo raíz instalado. El valor predeterminado es la ruta de acceso absoluta del archivo de catálogo raíz instalado (<carpeta-instalación>\Altova\RaptorXMLServer2024\etc\RootCatalog.xml). Consulte el apartado Catálogos XML [48] para obtener más información.

  - user-catalog

    --user-catalog = *ARCHIVO*

 © 2017-2023 Altova GmbH

Especifica la ruta de acceso absoluta a un catálogo XML que debe utilizarse junto con el catálogo raíz. Consulte el apartado Catálogos XML [48] para obtener más información.

- enable-globalresources

  --enable-globalresources = `true|false`
  Habilita la función de recursos globales [55]. Valor predeterminado: `false`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- globalresourceconfig [gc]

  --gc | --globalresourceconfig = *VALOR*
  Especifica la configuración activa del recurso global [55] (y habilita los recursos globales) [55].

- globalresourcefile [gr]

  --gr | --globalresourcefile = *ARCHIVO*
  Especifica el archivo de recursos globales [55] (y habilita los recursos globales) [55].

- Extensiones

  Estas opciones definen cómo se gestionan las funciones de extensión especiales disponibles en la edición Enterprise Edition de varios productos de Altova (como XMLSpy Enterprise Edition). Su uso se describe detalladamente en el manual del usuario de cada producto.

  - chartext-disable

    --chartext-disable = `true|false`
    Deshabilita las extensiones de gráficos. Valor predeterminado: `false`.
    Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

  - dotnetext-disable

    --dotnetext-disable = `true|false`
    Deshabilita las extensiones .NET. Valor predeterminado: `false`.
    Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

  - jvm-location

    --jvm-location = *ARCHIVO*
    ARCHIVO especifica la ubicación del equipo virtual Java (*DLL* en Windows, objeto *compartido (shared object)* en Linux. El JVM se necesita si usa las funciones de extensión Java [512] en su código XSLT/XQuery. El valor predeterminado es `false`.

  - javaext-barcode-location

    --javaext-barcode-location = *ARCHIVO*
    Especifica la ruta de acceso de la carpeta que contiene el archivo de extensión de código de barras `AltovaBarcodeExtension.jar`. La ruta de acceso debe darse en uno de estos formatos:

    - Un URI de archivo (ejemplo: `--javaext-barcode-location="file:///C:/Archivos de`

programa/Altova/RaptorXMLServer2024/etc/jar/")
- Una ruta de acceso Windows con caracteres de escape para las barras diagonales inversas (ejemplo: `--javaext-barcode-location="C:\\Archivos de programa\\Altova\ \RaptorXMLServer2024\\etc\\jar\\"`)

▼ javaext-disable

`--javaext-disable =` `true|false`
Deshabilita las extensiones Java. Valor predeterminado: `false`.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

▼ Opciones comunes

▼ error-format

`--error-format =` `text|shortxml|longxml`
Especifica el formato de la salida de error. Los valores posibles son formatos de texto, XML y XML detallado (`longxml`). Valor predeterminado: `text`.

▼ error-limit

`--error-limit =` *`N | unlimited`*
Especifica el límite de errores con un valor comprendido entre `1` y `9999` o `unlimited` (ilimitado). El valor predeterminado es `100`. Cuando se alcanza el límite de error, se detiene la validación. Esta opción es muy práctica a la hora de limitar el uso del procesador durante la validación o transformación.

▼ info-limit

`--info-limit =` *`N | unlimited`*
Indica el límite del mensaje de información dentro del rango `1-65535` or `unlimited`. Si se alcanza el límite de información indicado, el procesamiento continúa pero ya no se informa de más mensajes. El valor predeterminado es `100`.

▼ help

`--help`
Muestra el texto de ayuda para el comando. Por ejemplo `valany --h`. (Otra opción es usar el comando `help` con un argumento. Por ejemplo: `help valany`).

▼ listfile

`--listfile =` `true|false`
Si el valor es `true`, el argumento *`ArchivoEntrada`* del comando se entiende como un archivo de texto que contiene un nombre de archivo por línea. Otra opción es enumerar los archivos en la ILC, separados por un espacio. No obstante, recuerde que las ILC tienen un límite de caracteres. Además, no olvide que la opción `--listfile` solamente afecta a los argumentos y no a las opciones.
Valor predeterminado: `false`.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

 © 2017-2023 Altova GmbH

- log-output

  --log-output = *ARCHIVO*
  Escribe el registro de salida en la URL de archivo indicada. Compruebe que la ILC tiene permiso de escritura en la ubicación de destino.

- network-timeout

  --network-timeout = *VALOR*
  Indica el tiempo de espera en milisegundos para operaciones remotas de entrada y salida. Valor predeterminado: `40000`.

- recurse

  --recurse = `true|false`
  Esta opción se utiliza para seleccionar ficheros dentro de subdirectorios (incluso en archivos ZIP). Si el valor es `true`, el argumento *ArchivoEntrada* del comando seleccionará el fichero seleccionado también en los subdirectorios. Por ejemplo: `"test.zip|zip\test.xml"` seleccionará los ficheros llamados `test.xml` en todos los subdirectorios de la carpeta ZIP. Si quiere puede usar los caracteres comodín `*` y `?`. Por ejemplo: `*.xml` seleccionaría todos los ficheros de la carpeta ZIP que tengan la extensión `.xml`. Valor predeterminado: `false`
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- verbose

  --verbose = `true|false`
  Si el valor es `true`, se genera información adicional durante la validación. Valor predeterminado es `false`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- verbose-output

  --verbose-output = *ARCHIVO*
  Escribe el resultado detallado en el *ARCHIVO* indicado.

- version

  --version
  Muestra el número de versión de RaptorXML Server. Si se utiliza con un comando, escriba la opción `--version` antes del comando.

- warning-limit

  --warning-limit = *N | unlimited*
  Especifica el límite de advertencia en el rango `1-65535` o `unlimited` (ilimitado). El procesamiento continua si se alcanza el límite pero no se registrarán más advertencias. El valor por defecto es `100`.

5.5 Comandos JSON/Avro

Los comandos JSON sirven para comprobar la validez de esquemas y documentos de instancia JSON y comprobar si su formato es correcto. Para más información sobre cada comando consulte los apartados de esta sección:

- avroextractschema 144: Extrae el esquema Avro de un archivo binario Avro.
- json2xml 148: Convierte un documento de instancia JSON a un documento de instancia XML.
- jsonschema2xsd 153: Convierte un documento de esquema JSON a un documento de esquema XML.
- valavro 158: Valida los datos de archivos binarios Avro con el esquema Avro correspondiente de cada binario.
- valavrojson 162: Valida archivos de datos JSON con un esquema Avro.
- valavroschema 166: Valida un esquema Avro con la especificación Avro.
- valjsonschema 170: Comprueba la validez de documentos de esquema JSON.
- valjson 174: Comprueba la validez de documentos JSON.
- wfjson 179: Comprueba si el formato de los documentos JSON es correcto.
- xml2json 184: Convierte un documento de instancia XML a un documento de instancia JSON.
- xsd2jsonschema 189: Convierte un documento de esquema XML a un documento de esquema JSON.

5.5.1 avroextractschema

Un archivo binario Avro contiene un bloque de datos Avro precedido por un esquema Avro que define la estructura de dicho bloque de datos. El comando **avroextractschema** extrae el esquema Avro del binario Avro y lo serializa como JSON.

Windows	**RaptorXML avroextractschema [opciones] --avrooutput=*ArchivoEsquemaAvro* *ArchivoBinarioAvro***
Linux	**raptorxml avroextractschema [opciones] --avrooutput=*ArchivoEsquemaAvro* *ArchivoBinarioAvro***
Mac	**raptorxml avroextractschema [opciones] --avrooutput=*ArchivoEsquemaAvro* *ArchivoBinarioAvro***

El argumento *ArchivoBinarioAvro* especifica de qué archivo binario Avro se debe extraer el esquema Avro. La opción `--avrooutput` especifica la ubicación del esquema Avro que se extrae.

Ejemplos

```
raptorxml avroextractschema --avrooutput=c:\MiEsquemaAvro.avsc c:\MiBinarioAvro.avro
```

▼ Mayúsculas/minúsculas y barras en la línea de comandos

RaptorXML (y **RaptorXMLServer** para comandos administrativos) *en Windows*
raptorxml (y **raptorxmlserver** para comandos administrativos) *en Windows y Unix (Linux, Mac)*

 © 2017-2023 Altova GmbH

* Las minúsculas (`raptorxml` y `raptorxmlserver`) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que las mayúsculas (`RaptorXML`) funcionan solamente en Windows y Mac.
* Use la barra diagonal en Linux y Mac y la barra diagonal inversa en Windows.

▼ Barra diagonal inversa y espacios en sistemas Windows

En sistemas Windows: si hay espacios o caracteres especiales (por ejemplo en los nombres de archivos o de carpetas, empresas, personas o productos) debe usar comillas: por ejemplo, **`"Mi archivo"`**. Sin embargo, debe tener en cuenta que una barra diagonal inversa seguida de comillas dobles (por ejemplo, **`"C:\Mi Directorio\"`**) es posible que no se lea correctamente. Esto se debe a que la barra diagonal inversa también se usa para empezar secuencias de escape y la secuencia de escape **`\"`** también usa las comillas dobles. Para evitar secuencia de caracteres puede añadir otra barra diagonal inversa: **`\\"`**. En resumen: si necesita escribir una ruta que contenga espacios y una barra diagonal inversa, esta es la mejor forma de hacerlo: **`"C:\Mi Directorio\\"`**.

Opciones

Las opciones del comando aparecen más abajo divididas en grupos. Los valores se pueden dar sin comillas excepto en estos dos casos: (i) cuando la cadena de valor contiene espacios y (ii) cuando en la descripción de la opción se indique explícitamente que es necesario el uso de comillas. Si una opción toma un valor booleano y no se indica ningún valor, entonces el valor predeterminado de la opción es `TRUE`. Use la opción **`--h, --help`** para ver información sobre el comando.

▼ Procesamiento

▼ output, **avrooutput**

`--output =` *`ARCHIVO`*, **`--avrooutput =`** *`ARCHIVO`*
Establece la ubicación del archivo Avro de salida.

▼ recurse

`--recurse =` `true|false`
Esta opción se utiliza para seleccionar ficheros dentro de subdirectorios (incluso en archivos ZIP). Si el valor es `true`, el argumento *`ArchivoEntrada`* del comando seleccionará el fichero seleccionado también en los subdirectorios. Por ejemplo: `"test.zip|zip\test.xml"` seleccionará los ficheros llamados `test.xml` en todos los subdirectorios de la carpeta ZIP. Si quiere puede usar los caracteres comodín `*` y `?`. Por ejemplo: `*.xml` seleccionaría todos los ficheros de la carpeta ZIP que tengan la extensión `.xml`. Valor predeterminado: `false`
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

▼ Catálogos y recursos globales

▼ catalog

`--catalog =` *`ARCHIVO`*
Especifica la ruta de acceso absoluta a un archivo de catálogo que no está en el archivo de catálogo raíz instalado. El valor predeterminado es la ruta de acceso absoluta del archivo de catálogo raíz instalado (`<carpeta-instalación>\Altova\RaptorXMLServer2024\etc\RootCatalog.xml`). Consulte el apartado Catálogos XML [48] para obtener más información.

- user-catalog

  --user-catalog = *ARCHIVO*
  Especifica la ruta de acceso absoluta a un catálogo XML que debe utilizarse junto con el catálogo raíz. Consulte el apartado Catálogos XML [48] para obtener más información.

- enable-globalresources

  --enable-globalresources = `true|false`
  Habilita la función de recursos globales [55]. Valor predeterminado: `false`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- globalresourceconfig [gc]

  --gc | --globalresourceconfig = *VALOR*
  Especifica la configuración activa del recurso global [55] (y habilita los recursos globales) [55].

- globalresourcefile [gr]

  --gr | --globalresourcefile = *ARCHIVO*
  Especifica el archivo de recursos globales [55] (y habilita los recursos globales) [55].

- Opciones comunes
  - error-format

    --error-format = `text|shortxml|longxml`
    Especifica el formato de la salida de error. Los valores posibles son formatos de texto, XML y XML detallado (`longxml`). Valor predeterminado: `text`.

  - error-limit

    --error-limit = *N | unlimited*
    Especifica el límite de errores con un valor comprendido entre `1` y `9999` o `unlimited` (ilimitado). El valor predeterminado es `100`. Cuando se alcanza el límite de error, se detiene la validación. Esta opción es muy práctica a la hora de limitar el uso del procesador durante la validación o transformación.

  - info-limit

    --info-limit = *N | unlimited*
    Indica el límite del mensaje de información dentro del rango `1-65535` or `unlimited`. Si se alcanza el límite de información indicado, el procesamiento continúa pero ya no se informa de más mensajes. El valor predeterminado es `100`.

  - help

    --help
    Muestra el texto de ayuda para el comando. Por ejemplo `valany --h`. (Otra opción es usar el comando `help` con un argumento. Por ejemplo: `help valany`).

 © 2017-2023 Altova GmbH

- listfile

  --listfile = `true|false`
  Si el valor es `true`, el argumento *`ArchivoEntrada`* del comando se entiende como un archivo de texto que contiene un nombre de archivo por línea. Otra opción es enumerar los archivos en la ILC, separados por un espacio. No obstante, recuerde que las ILC tienen un límite de caracteres. Además, no olvide que la opción `--listfile` solamente afecta a los argumentos y no a las opciones.
  Valor predeterminado: `false`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- log-output

  --log-output = *`ARCHIVO`*
  Escribe el registro de salida en la URL de archivo indicada. Compruebe que la ILC tiene permiso de escritura en la ubicación de destino.

- network-timeout

  --network-timeout = *`VALOR`*
  Indica el tiempo de espera en milisegundos para operaciones remotas de entrada y salida. Valor predeterminado: `40000`.

- recurse

  --recurse = `true|false`
  Esta opción se utiliza para seleccionar ficheros dentro de subdirectorios (incluso en archivos ZIP). Si el valor es `true`, el argumento *`ArchivoEntrada`* del comando seleccionará el fichero seleccionado también en los subdirectorios. Por ejemplo: `"test.zip|zip\test.xml"` seleccionará los ficheros llamados `test.xml` en todos los subdirectorios de la carpeta ZIP. Si quiere puede usar los caracteres comodín `*` y `?`. Por ejemplo: `*.xml` seleccionaría todos los ficheros de la carpeta ZIP que tengan la extensión `.xml`. Valor predeterminado: `false`
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- verbose

  --verbose = `true|false`
  Si el valor es `true`, se genera información adicional durante la validación. Valor predeterminado es `false`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- verbose-output

  --verbose-output = *`ARCHIVO`*
  Escribe el resultado detallado en el *`ARCHIVO`* indicado.

- version

  --version
  Muestra el número de versión de RaptorXML Server. Si se utiliza con un comando, escriba la opción `--version` antes del comando.

- warning-limit

  --warning-limit = *N | unlimited*
  Especifica el límite de advertencia en el rango `1-65535` o `unlimited` (ilimitado). El procesamiento continua si se alcanza el límite pero no se registrarán más advertencias. El valor por defecto es `100`.

5.5.2 json2xml

El comando **json2xml** convierte un documento de instancia JSON en un documento de instancia XML.

```
raptorxml json2xml [opciones] ArchivoJSON
```

- El argumento *ArchivoJSON* es el documento JSON que se quiere convertir.
- Utilice la opción `--conversion-output` para especificar dónde se debe guardar el archivo XML generado.

Ejemplo

Ejemplo del comando **json2xml**:

- **raptorxml** `json2xml --conversion-output=c:\MisDatosXML.xml c:\MisDatosJSON.json`

- Mayúsculas/minúsculas y barras en la línea de comandos

  RaptorXML (y **RaptorXMLServer** para comandos administrativos) *en Windows*
  raptorxml (y **raptorxmlserver** para comandos administrativos) *en Windows y Unix (Linux, Mac)*

  * Las minúsculas (`raptorxml` y `raptorxmlserver`) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que las mayúsculas (`RaptorXML`) funcionan solamente en Windows y Mac.
  * Use la barra diagonal en Linux y Mac y la barra diagonal inversa en Windows.

- Barra diagonal inversa y espacios en sistemas Windows

  En sistemas Windows: si hay espacios o caracteres especiales (por ejemplo en los nombres de archivos o de carpetas, empresas, personas o productos) debe usar comillas: por ejemplo, **"Mi archivo"**. Sin embargo, debe tener en cuenta que una barra diagonal inversa seguida de comillas dobles (por ejemplo, **"C:\Mi Directorio\"**) es posible que no se lea correctamente. Esto se debe a que la barra diagonal inversa también se usa para empezar secuencias de escape y la secuencia de escape **\"** también usa las comillas dobles. Para evitar secuencia de caracteres puede añadir otra barra diagonal inversa: **\\"**. En resumen: si necesita escribir una ruta que contenga espacios y una barra diagonal inversa, esta es la mejor forma de hacerlo: **"C:\Mi Directorio\\"**.

© 2017-2023 Altova GmbH

Opciones

Las opciones del comando aparecen más abajo divididas en grupos. Los valores se pueden dar sin comillas excepto en estos dos casos: (i) cuando la cadena de valor contiene espacios y (ii) cuando en la descripción de la opción se indique explícitamente que es necesario el uso de comillas. Si una opción toma un valor booleano y no se indica ningún valor, entonces el valor predeterminado de la opción es `TRUE`. Use la opción **`--h, --help`** para ver información sobre el comando.

Opciones de conversión de datos JSON en XML

Estas opciones definen cómo se gestionan los detalles específicos relacionados con la conversión de datos XML en JSON.

array-element

`--array-element =` `VALUE`
Especifica el nombre del elemento que se convertirá en un elemento de matriz.

attributes

`--attributes =` `true|false`
Si está establecido en **`true`**, se activa la conversión entre atributos XML y propiedades JSON prefijadas con @. Si no está activada esta opción, no se convertirán ni atributos XML ni propiedades JSON con el prefijo @. El valor predeterminado para esta opción es **`true`**.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

comments

`--comments =` `true|false`
Si está establecido en **`true`**, se activa la conversión entre comentarios XML y propiedades JSON prefijadas con #. Si no está activada esta opción, no se convertirán ni atributos XML ni propiedades JSON con el prefijo #. El valor predeterminado para esta opción es **`true`**.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

conversion-output, o

`--o, --conversion-output =` `FILE`
Establece el nombre y la ruta de acceso del archivo al que se envía el resultado de la conversión.

create-array-container

`--create-array-container =` `true|false`
Si está establecido en **`true`**, se crea un elemento de contenedor en el archivo XML generado para cada matriz JSON en el documento JSON de origen. El valor predeterminado para esta opción es **`false`**.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

encode-colons

`--encode-colons =` `true|false`
Si está establecido en **`true`**, en el documento XML generado se codifican los dos puntos en los

nombres de propiedades JSON. El valor predeterminado para esta opción es **true**.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- json-type-hints

  --json-type-hints = `true|false`
  Si está establecido en **true**, se añaden atributos al documento XML generado para las sugerencias de tipo en el documento JSON de origen. El valor predeterminado para esta opción es **true**.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- pi

  --pi = `true|false`
  Si está establecido en **true**, se activa la conversión entre instrucciones de procesamiento XML y propiedades JSON prefijadas con ?. Si no está activada esta opción, no se convertirán ni atributos XML ni propiedades JSON con el prefijo ?. El valor predeterminado para esta opción es **true**.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- pretty-print

  --pp, --pretty-print = `true|false`
  Si está establecido en **true**, se genera un pretty-print del documento de salida generado. El valor predeterminado para esta opción es **false**.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- text

  --text = `true|false`
  Si está establecido en **true**, se activa la conversión entre contenido de texto XML y propiedades JSON prefijadas con $. Si no está activada esta opción, no se convertirán ni atributos XML ni propiedades JSON con el prefijo $. El valor predeterminado para esta opción es **true**.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- Opciones comunes
  - error-format

    --error-format = `text|shortxml|longxml`
    Especifica el formato de la salida de error. Los valores posibles son formatos de texto, XML y XML detallado (`longxml`). Valor predeterminado: `text`.

  - error-limit

    --error-limit = *`N | unlimited`*
    Especifica el límite de errores con un valor comprendido entre `1` y `9999` o `unlimited` (ilimitado). El valor predeterminado es `100`. Cuando se alcanza el límite de error, se detiene la validación. Esta opción es muy práctica a la hora de limitar el uso del procesador durante la validación o

 © 2017-2023 Altova GmbH

transformación.

- info-limit

  --info-limit = *N | unlimited*
  Indica el límite del mensaje de información dentro del rango `1-65535` or `unlimited`. Si se alcanza el límite de información indicado, el procesamiento continúa pero ya no se informa de más mensajes. El valor predeterminado es `100`.

- help

  --help
  Muestra el texto de ayuda para el comando. Por ejemplo `valany --h`. (Otra opción es usar el comando `help` con un argumento. Por ejemplo: `help valany`).

- listfile

  --listfile = `true|false`
  Si el valor es `true`, el argumento *ArchivoEntrada* del comando se entiende como un archivo de texto que contiene un nombre de archivo por línea. Otra opción es enumerar los archivos en la ILC, separados por un espacio. No obstante, recuerde que las ILC tienen un límite de caracteres. Además, no olvide que la opción `--listfile` solamente afecta a los argumentos y no a las opciones.
  Valor predeterminado: `false`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- log-output

  --log-output = *ARCHIVO*
  Escribe el registro de salida en la URL de archivo indicada. Compruebe que la ILC tiene permiso de escritura en la ubicación de destino.

- network-timeout

  --network-timeout = *VALOR*
  Indica el tiempo de espera en milisegundos para operaciones remotas de entrada y salida. Valor predeterminado: `40000`.

- recurse

  --recurse = `true|false`
  Esta opción se utiliza para seleccionar ficheros dentro de subdirectorios (incluso en archivos ZIP). Si el valor es `true`, el argumento *ArchivoEntrada* del comando seleccionará el fichero seleccionado también en los subdirectorios. Por ejemplo: `"test.zip|zip\test.xml"` seleccionará los ficheros llamados `test.xml` en todos los subdirectorios de la carpeta ZIP. Si quiere puede usar los caracteres comodín `*` y `?`. Por ejemplo: `*.xml` seleccionaría todos los ficheros de la carpeta ZIP que tengan la extensión `.xml`. Valor predeterminado: `false`
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- verbose

  --verbose = `true|false`

Si el valor es `true`, se genera información adicional durante la validación. Valor predeterminado es `false`.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- verbose-output

  `--verbose-output =` *`ARCHIVO`*
  Escribe el resultado detallado en el *`ARCHIVO`* indicado.

- version

  `--version`
  Muestra el número de versión de RaptorXML Server. Si se utiliza con un comando, escriba la opción `--version` antes del comando.

- warning-limit

  `--warning-limit =` *`N | unlimited`*
  Especifica el límite de advertencia en el rango `1-65535` o `unlimited` (ilimitado). El procesamiento continua si se alcanza el límite pero no se registrarán más advertencias. El valor por defecto es `100`.

- Catálogos y recursos globales
  - catalog

    `--catalog =` *`ARCHIVO`*
    Especifica la ruta de acceso absoluta a un archivo de catálogo que no está en el archivo de catálogo raíz instalado. El valor predeterminado es la ruta de acceso absoluta del archivo de catálogo raíz instalado (`<carpeta-instalación>\Altova\RaptorXMLServer2024\etc\RootCatalog.xml`). Consulte el apartado Catálogos XML [48] para obtener más información.

  - user-catalog

    `--user-catalog =` *`ARCHIVO`*
    Especifica la ruta de acceso absoluta a un catálogo XML que debe utilizarse junto con el catálogo raíz. Consulte el apartado Catálogos XML [48] para obtener más información.

  - enable-globalresources

    `--enable-globalresources =` `true|false`
    Habilita la función de recursos globales [55]. Valor predeterminado: `false`.
    Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

  - globalresourceconfig [gc]

    `--gc | --globalresourceconfig =` *`VALOR`*
    Especifica la configuración activa del recurso global [55] (y habilita los recursos globales) [55].

  - globalresourcefile [gr]

    `--gr | --globalresourcefile =` *`ARCHIVO`*

© 2017-2023 Altova GmbH

Especifica el archivo de recursos globales 55 (y habilita los recursos globales) 55.

5.5.3 jsonschema2xsd

El comando **jsonschema2xsd** convierte un documento de esquema JSON en un documento de esquema XML que sea conforme a las especificaciones 1.0 y 1.1 del XSD del W3C.

raptorxml jsonschema2xsd [opciones] ***ArchivoEsquemaJSON***

- El argumento *ArchivoEsquemaJSON* es el archivo de esquema JSON que se quiere convertir.
- Utilice la opción `--schema-conversion-output` para especificar dónde se debe guardar el archivo XSD generado.

Ejemplo

Ejemplo del comando **jsonschema2xsd**:

- **raptorxml** jsonschema2xsd --schema-conversion-output=c:\MiEsquemaXML.xsd c:\MiEsquemaJSON.json

▼ Mayúsculas/minúsculas y barras en la línea de comandos

RaptorXML (y **RaptorXMLServer** para comandos administrativos) *en Windows*
raptorxml (y **raptorxmlserver** para comandos administrativos) *en Windows y Unix (Linux, Mac)*

* Las minúsculas (`raptorxml` y `raptorxmlserver`) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que las mayúsculas (`RaptorXML`) funcionan solamente en Windows y Mac.
* Use la barra diagonal en Linux y Mac y la barra diagonal inversa en Windows.

▼ Barra diagonal inversa y espacios en sistemas Windows

En sistemas Windows: si hay espacios o caracteres especiales (por ejemplo en los nombres de archivos o de carpetas, empresas, personas o productos) debe usar comillas: por ejemplo, **"Mi archivo"**. Sin embargo, debe tener en cuenta que una barra diagonal inversa seguida de comillas dobles (por ejemplo, **"C:\Mi Directorio\"**) es posible que no se lea correctamente. Esto se debe a que la barra diagonal inversa también se usa para empezar secuencias de escape y la secuencia de escape **\"** también usa las comillas dobles. Para evitar secuencia de caracteres puede añadir otra barra diagonal inversa: **\\"**. En resumen: si necesita escribir una ruta que contenga espacios y una barra diagonal inversa, esta es la mejor forma de hacerlo: **"C:\Mi Directorio\\"**.

Opciones

Las opciones del comando aparecen más abajo divididas en grupos. Los valores se pueden dar sin comillas excepto en estos dos casos: (i) cuando la cadena de valor contiene espacios y (ii) cuando en la descripción de

la opción se indique explícitamente que es necesario el uso de comillas. Si una opción toma un valor booleano y no se indica ningún valor, entonces el valor predeterminado de la opción es `TRUE`. Use la opción **`--h, --help`** para ver información sobre el comando.

Opciones de validación JSON

Estas son las opciones para validar el documento de esquema JSON de entrada.

additional-schema

`--additional-schema =` *FILE*

Indica los URIs de un documento de esquema suplementario. El esquema suplementario se carga desde el principal y se puede hacer referencia a él también desde el esquema principal con la propiedad **`id`** o **`$id`** del esquema suplementario.

disable-format-checks

`--disable-format-checks =` `true|false`

Deshabilita la validación semántica impuesta por el atributo de formato. El valor predeterminado es `false`.

Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

jsonschema-version

`--jsonschema-version =` `draft04|draft06|draft07|2019-09|2020-12|latest|detect`

Indica qué versión de la versión del borrador de la especificación JSON Schema usar. El valor predeterminado es `detect`.

strict-integer-checks

`--strict-integer-checks =` `true|false`

Indica si se debe usar la comprobación estricta de números enteros del borrador draft-04 con esquemas posteriores donde las comprobaciones de números enteros son más laxas. Por ejemplo, **`1.0`** no es un número entero válido en draft-04 pero sí en borradores posteriores. Esta opción no afecta a los esquemas de draft-04. El valor predeterminado de esta opción es `false`.

Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

Conversión de esquema JSON a archivo XSD

Estas con las opciones para especificar los detalles de la conversión de esquemas JSON en archivos XSD.

at-to-attributes

`--at-to-attributes =` `true|false`

Si está establecido en **`true`**, las propiedades con prefijo @ en el documento de esquema JSON se convierten en atributos en el documento XSD generado. El valor predeterminado para esta opción es **`true`**.

Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

dollar-to-text

`--dollar-to-text =` `true|false`

 © 2017-2023 Altova GmbH

Si está establecido en **true**, las propiedades con prefijo $ en el documento de esquema JSON se convierten en texto en el documento XSD generado. El valor predeterminado para esta opción es **true**.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- ignore-comments

  --ignore-comments = `true|false`
  Si está configurado en **true**, se omiten las propiedades llamadas `'#'` en el esquema JSON de origen. El valor predeterminado para esta opción es **true**.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- ignore-pi-proprties

  --ignore-pi-properties = `true|false`
  Si está configurado en **true**, se omiten las propiedades que empiezan con `'?'` en el esquema JSON de origen. El valor predeterminado para esta opción es **true**.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- ignore-xmlns-proprties

  --ignore-xmlns-properties = `true|false`
  Si está configurado en **true**, se omiten las propiedades que empiezan con `'@xmlns'` en el esquema JSON de origen. El valor predeterminado para esta opción es **true**.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- ignore-xsi-proprties

  --ignore-xsi-properties = `true|false`
  Si está configurado en **true**, se omiten las propiedades que empiezan con `'@ xsi'` en el esquema JSON de origen. El valor predeterminado para esta opción es **true**.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- schema-conversion-output, o

  --o, --schema-converson-output = `FILE`
  Establece el nombre y la ruta de acceso del archivo al que se envía el resultado de la conversión.

- Opciones comunes
  - error-format

    --error-format = `text|shortxml|longxml`
    Especifica el formato de la salida de error. Los valores posibles son formatos de texto, XML y XML detallado (`longxml`). Valor predeterminado: `text`.

  - error-limit

--error-limit = *N | unlimited*
Especifica el límite de errores con un valor comprendido entre `1` y `9999` o `unlimited` (ilimitado). El valor predeterminado es `100`. Cuando se alcanza el límite de error, se detiene la validación. Esta opción es muy práctica a la hora de limitar el uso del procesador durante la validación o transformación.

- info-limit

--info-limit = *N | unlimited*
Indica el límite del mensaje de información dentro del rango `1-65535` or `unlimited`. Si se alcanza el límite de información indicado, el procesamiento continúa pero ya no se informa de más mensajes. El valor predeterminado es `100`.

- help

--help
Muestra el texto de ayuda para el comando. Por ejemplo `valany --h`. (Otra opción es usar el comando `help` con un argumento. Por ejemplo: `help valany`).

- listfile

--listfile = `true|false`
Si el valor es `true`, el argumento *ArchivoEntrada* del comando se entiende como un archivo de texto que contiene un nombre de archivo por línea. Otra opción es enumerar los archivos en la ILC, separados por un espacio. No obstante, recuerde que las ILC tienen un límite de caracteres. Además, no olvide que la opción `--listfile` solamente afecta a los argumentos y no a las opciones.
Valor predeterminado: `false`.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- log-output

--log-output = *ARCHIVO*
Escribe el registro de salida en la URL de archivo indicada. Compruebe que la ILC tiene permiso de escritura en la ubicación de destino.

- network-timeout

--network-timeout = *VALOR*
Indica el tiempo de espera en milisegundos para operaciones remotas de entrada y salida. Valor predeterminado: `40000`.

- recurse

--recurse = `true|false`
Esta opción se utiliza para seleccionar ficheros dentro de subdirectorios (incluso en archivos ZIP). Si el valor es `true`, el argumento *ArchivoEntrada* del comando seleccionará el fichero seleccionado también en los subdirectorios. Por ejemplo: `"test.zip|zip\test.xml"` seleccionará los ficheros llamados `test.xml` en todos los subdirectorios de la carpeta ZIP. Si quiere puede usar los caracteres comodín `*` y `?`. Por ejemplo: `*.xml` seleccionaría todos los ficheros de la carpeta ZIP que tengan la extensión `.xml`. Valor predeterminado: `false`
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en

 © 2017-2023 Altova GmbH

true.

- verbose

  --verbose = true|false
  Si el valor es true, se genera información adicional durante la validación. Valor predeterminado es false.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en true.

- verbose-output

  --verbose-output = *ARCHIVO*
  Escribe el resultado detallado en el *ARCHIVO* indicado.

- version

  --version
  Muestra el número de versión de RaptorXML Server. Si se utiliza con un comando, escriba la opción --version antes del comando.

- warning-limit

  --warning-limit = *N | unlimited*
  Especifica el límite de advertencia en el rango 1-65535 o unlimited (ilimitado). El procesamiento continua si se alcanza el límite pero no se registrarán más advertencias. El valor por defecto es 100.

- Catálogos y recursos globales
  - catalog

    --catalog = *ARCHIVO*
    Especifica la ruta de acceso absoluta a un archivo de catálogo que no está en el archivo de catálogo raíz instalado. El valor predeterminado es la ruta de acceso absoluta del archivo de catálogo raíz instalado (<carpeta-instalación>\Altova\RaptorXMLServer2024\etc\RootCatalog.xml). Consulte el apartado Catálogos XML [48] para obtener más información.

  - user-catalog

    --user-catalog = *ARCHIVO*
    Especifica la ruta de acceso absoluta a un catálogo XML que debe utilizarse junto con el catálogo raíz. Consulte el apartado Catálogos XML [48] para obtener más información.

  - enable-globalresources

    --enable-globalresources = true|false
    Habilita la función de recursos globales [55]. Valor predeterminado: false.
    Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en true.

  - globalresourceconfig [gc]

    --gc | --globalresourceconfig = *VALOR*

Especifica la configuración activa del recurso global[55] (y habilita los recursos globales)[55].

- globalresourcefile [gr]

  --gr | --globalresourcefile = *ARCHIVO*
  Especifica el archivo de recursos globales[55] (y habilita los recursos globales)[55].

5.5.4 valavro (avro)

El comando **valavro | avro** valida los bloques de datos de archivos binarios con los esquemas Avro correspondientes de cada archivo binario.

Windows	**RaptorXML valavro	avro [opciones]** ***ArchivoBinarioAvro***
Linux	**raptorxml valavro	avro [opciones]** ***ArchivoBinarioAvro***
Mac	**raptorxml valavro	avro [opciones]** ***ArchivoBinarioAvro***

El argumento *ArchivoBinarioAvro* especifica qué archivos binarios Avro se deben validar. Concretamente, el bloque de datos de cada archivo binario Avro se valida con el esquema Avro del archivo correspondiente.

Si desea validar varios binarios Avro, tiene dos opciones: (i) enumerar los archivos que se deben validar en la línea de comandos, separados por espacios o (ii) enumerar los archivos que se deben validar en un archivo de texto (`.txt`) donde aparece un nombre de archivo por línea y dar este archivo de texto como argumento *ArchivoBinarioAvro* junto con la opción `--listfile`[238] con valor `true` (*ver lista de opciones más abajo*).

Ejemplos

- **raptorxml** valavro c:\MiBinarioAvro.avro
- **raptorxml** valavro c:\MiBinarioAvro01.avro c:\MiBinarioAvro02.avro
- **raptorxml** avro--listfile=true c:\MiListaArchivos.txt

- Mayúsculas/minúsculas y barras en la línea de comandos

  RaptorXML (y **RaptorXMLServer** para comandos administrativos) *en Windows*
  raptorxml (y **raptorxmlserver** para comandos administrativos) *en Windows y Unix (Linux, Mac)*

  * Las minúsculas (`raptorxml` y `raptorxmlserver`) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que las mayúsculas (`RaptorXML`) funcionan solamente en Windows y Mac.
  * Use la barra diagonal en Linux y Mac y la barra diagonal inversa en Windows.

- Barra diagonal inversa y espacios en sistemas Windows

  En sistemas Windows: si hay espacios o caracteres especiales (por ejemplo en los nombres de archivos

 © 2017-2023 Altova GmbH

o de carpetas, empresas, personas o productos) debe usar comillas: por ejemplo, **"Mi archivo"**. Sin embargo, debe tener en cuenta que una barra diagonal inversa seguida de comillas dobles (por ejemplo, **"C:\Mi Directorio\"**) es posible que no se lea correctamente. Esto se debe a que la barra diagonal inversa también se usa para empezar secuencias de escape y la secuencia de escape **\"** también usa las comillas dobles. Para evitar secuencia de caracteres puede añadir otra barra diagonal inversa: **\\"**. En resumen: si necesita escribir una ruta que contenga espacios y una barra diagonal inversa, esta es la mejor forma de hacerlo: **"C:\Mi Directorio\\"**.

Opciones

Las opciones del comando aparecen más abajo divididas en grupos. Los valores se pueden dar sin comillas excepto en estos dos casos: (i) cuando la cadena de valor contiene espacios y (ii) cuando en la descripción de la opción se indique explícitamente que es necesario el uso de comillas. Si una opción toma un valor booleano y no se indica ningún valor, entonces el valor predeterminado de la opción es `TRUE`. Use la opción **--h, --help** para ver información sobre el comando.

- Procesamiento
  - listfile

    --listfile = `true|false`
    Si el valor es `true`, el argumento *ArchivoEntrada* del comando se entiende como un archivo de texto que contiene un nombre de archivo por línea. Otra opción es enumerar los archivos en la ILC, separados por un espacio. No obstante, recuerde que las ILC tienen un límite de caracteres. Además, no olvide que la opción `--listfile` solamente afecta a los argumentos y no a las opciones.
    Valor predeterminado: `false`.
    Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

  - recurse

    --recurse = `true|false`
    Esta opción se utiliza para seleccionar ficheros dentro de subdirectorios (incluso en archivos ZIP). Si el valor es `true`, el argumento *ArchivoEntrada* del comando seleccionará el fichero seleccionado también en los subdirectorios. Por ejemplo: `"test.zip|zip\test.xml"` seleccionará los ficheros llamados `test.xml` en todos los subdirectorios de la carpeta ZIP. Si quiere puede usar los caracteres comodín `*` y `?`. Por ejemplo: `*.xml` seleccionaría todos los ficheros de la carpeta ZIP que tengan la extensión `.xml`. Valor predeterminado: `false`
    Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- Catálogos y recursos globales
  - catalog

    --catalog = *ARCHIVO*
    Especifica la ruta de acceso absoluta a un archivo de catálogo que no está en el archivo de catálogo raíz instalado. El valor predeterminado es la ruta de acceso absoluta del archivo de catálogo raíz instalado (`<carpeta-instalación>\Altova\RaptorXMLServer2024\etc\RootCatalog.xml`). Consulte el apartado Catálogos XML [48] para obtener más información.

- user-catalog

  --user-catalog = *ARCHIVO*
  Especifica la ruta de acceso absoluta a un catálogo XML que debe utilizarse junto con el catálogo raíz. Consulte el apartado Catálogos XML [48] para obtener más información.

- enable-globalresources

  --enable-globalresources = `true|false`
  Habilita la función de recursos globales [55]. Valor predeterminado: `false`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- globalresourceconfig [gc]

  --gc | --globalresourceconfig = *VALOR*
  Especifica la configuración activa del recurso global [55] (y habilita los recursos globales) [55].

- globalresourcefile [gr]

  --gr | --globalresourcefile = *ARCHIVO*
  Especifica el archivo de recursos globales [55] (y habilita los recursos globales) [55].

- Opciones comunes
  - error-format

    --error-format = `text|shortxml|longxml`
    Especifica el formato de la salida de error. Los valores posibles son formatos de texto, XML y XML detallado (`longxml`). Valor predeterminado: `text`.

  - error-limit

    --error-limit = *N | unlimited*
    Especifica el límite de errores con un valor comprendido entre `1` y `9999` o `unlimited` (ilimitado). El valor predeterminado es `100`. Cuando se alcanza el límite de error, se detiene la validación. Esta opción es muy práctica a la hora de limitar el uso del procesador durante la validación o transformación.

  - info-limit

    --info-limit = *N | unlimited*
    Indica el límite del mensaje de información dentro del rango `1-65535` or `unlimited`. Si se alcanza el límite de información indicado, el procesamiento continúa pero ya no se informa de más mensajes. El valor predeterminado es `100`.

  - help

    --help
    Muestra el texto de ayuda para el comando. Por ejemplo `valany --h`. (Otra opción es usar el comando `help` con un argumento. Por ejemplo: `help valany`).

 © 2017-2023 Altova GmbH

- listfile

  --listfile = `true|false`
  Si el valor es `true`, el argumento *`ArchivoEntrada`* del comando se entiende como un archivo de texto que contiene un nombre de archivo por línea. Otra opción es enumerar los archivos en la ILC, separados por un espacio. No obstante, recuerde que las ILC tienen un límite de caracteres. Además, no olvide que la opción `--listfile` solamente afecta a los argumentos y no a las opciones.
  Valor predeterminado: `false`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- log-output

  --log-output = *`ARCHIVO`*
  Escribe el registro de salida en la URL de archivo indicada. Compruebe que la ILC tiene permiso de escritura en la ubicación de destino.

- network-timeout

  --network-timeout = *`VALOR`*
  Indica el tiempo de espera en milisegundos para operaciones remotas de entrada y salida. Valor predeterminado: `40000`.

- recurse

  --recurse = `true|false`
  Esta opción se utiliza para seleccionar ficheros dentro de subdirectorios (incluso en archivos ZIP). Si el valor es `true`, el argumento *`ArchivoEntrada`* del comando seleccionará el fichero seleccionado también en los subdirectorios. Por ejemplo: `"test.zip|zip\test.xml"` seleccionará los ficheros llamados `test.xml` en todos los subdirectorios de la carpeta ZIP. Si quiere puede usar los caracteres comodín `*` y `?`. Por ejemplo: `*.xml` seleccionaría todos los ficheros de la carpeta ZIP que tengan la extensión `.xml`. Valor predeterminado: `false`
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- verbose

  --verbose = `true|false`
  Si el valor es `true`, se genera información adicional durante la validación. Valor predeterminado es `false`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- verbose-output

  --verbose-output = *`ARCHIVO`*
  Escribe el resultado detallado en el *`ARCHIVO`* indicado.

- version

  --version
  Muestra el número de versión de RaptorXML Server. Si se utiliza con un comando, escriba la opción `--version` antes del comando.

- warning-limit

  --warning-limit = *N | unlimited*
  Especifica el límite de advertencia en el rango `1-65535` o `unlimited` (ilimitado). El procesamiento continua si se alcanza el límite pero no se registrarán más advertencias. El valor por defecto es `100`.

5.5.5 valavrojson (avrojson)

El comando **valavrojson | avrojson** valida un documento JSON con un esquema Avro.

Windows	**RaptorXML valavrojson | avrojson [opciones] --avroschema=*EsquemaAvro ArchivoJSON***
Linux	**raptorxml valavrojson | avrojson [opciones] --avroschema=*EsquemaAvro ArchivoJSON***
Mac	**raptorxml valavrojson | avrojson [opciones] --avroschema=*EsquemaAvro ArchivoJSON***

El argumento *ArchivoJSON* especifica qué documento JSON se debe validar. La opción `--avroschema` especifica qué esquema Avro se debe usar para validar el documento JSON.

Si desea validar varios archivos JSON, tiene dos opciones: (i) enumerar los archivos que se deben validar en la línea de comandos, separados por espacios o (ii) enumerar los archivos que se deben validar en un archivo de texto (`.txt`) donde aparece un nombre de archivo por línea y dar este archivo de texto como argumento *ArchivoJSON* junto con la opción `--listfile` 238 con valor `true` (*ver lista de opciones más abajo*).

Ejemplos

- **raptorxml** `valavrojson --avroschema=c:\MiEsquemaAvro.avsc c:\MiArchivoJSON.json`
- **raptorxml** `avrojson --avroschema=c:\MiEsquemaAvro.avsc c:\MiArchivoJSON.json`

- Mayúsculas/minúsculas y barras en la línea de comandos

  RaptorXML (y **RaptorXMLServer** para comandos administrativos) *en Windows*
  raptorxml (y **raptorxmlserver** para comandos administrativos) *en Windows y Unix (Linux, Mac)*

  * Las minúsculas (`raptorxml` y `raptorxmlserver`) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que las mayúsculas (`RaptorXML`) funcionan solamente en Windows y Mac.
  * Use la barra diagonal en Linux y Mac y la barra diagonal inversa en Windows.

- Barra diagonal inversa y espacios en sistemas Windows

  En sistemas Windows: si hay espacios o caracteres especiales (por ejemplo en los nombres de archivos

 © 2017-2023 Altova GmbH

o de carpetas, empresas, personas o productos) debe usar comillas: por ejemplo, **"Mi archivo"**. Sin embargo, debe tener en cuenta que una barra diagonal inversa seguida de comillas dobles (por ejemplo, **"C:\Mi Directorio\"**) es posible que no se lea correctamente. Esto se debe a que la barra diagonal inversa también se usa para empezar secuencias de escape y la secuencia de escape **\"** también usa las comillas dobles. Para evitar secuencia de caracteres puede añadir otra barra diagonal inversa: **\\"**. En resumen: si necesita escribir una ruta que contenga espacios y una barra diagonal inversa, esta es la mejor forma de hacerlo: **"C:\Mi Directorio\\"**.

Opciones

Las opciones del comando aparecen más abajo divididas en grupos. Los valores se pueden dar sin comillas excepto en estos dos casos: (i) cuando la cadena de valor contiene espacios y (ii) cuando en la descripción de la opción se indique explícitamente que es necesario el uso de comillas. Si una opción toma un valor booleano y no se indica ningún valor, entonces el valor predeterminado de la opción es `TRUE`. Use la opción **`--h, --help`** para ver información sobre el comando.

- Procesamiento
  - listfile

    `--listfile =` `true|false`
    Si el valor es `true`, el argumento *`ArchivoEntrada`* del comando se entiende como un archivo de texto que contiene un nombre de archivo por línea. Otra opción es enumerar los archivos en la ILC, separados por un espacio. No obstante, recuerde que las ILC tienen un límite de caracteres. Además, no olvide que la opción `--listfile` solamente afecta a los argumentos y no a las opciones.
    Valor predeterminado: `false`.
    Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

  - recurse

    `--recurse =` `true|false`
    Esta opción se utiliza para seleccionar ficheros dentro de subdirectorios (incluso en archivos ZIP). Si el valor es `true`, el argumento *`ArchivoEntrada`* del comando seleccionará el fichero seleccionado también en los subdirectorios. Por ejemplo: `"test.zip|zip\test.xml"` seleccionará los ficheros llamados `test.xml` en todos los subdirectorios de la carpeta ZIP. Si quiere puede usar los caracteres comodín `*` y `?`. Por ejemplo: `*.xml` seleccionaría todos los ficheros de la carpeta ZIP que tengan la extensión `.xml`. Valor predeterminado: `false`
    Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- Catálogos y recursos globales
  - catalog

    `--catalog =` *`ARCHIVO`*
    Especifica la ruta de acceso absoluta a un archivo de catálogo que no está en el archivo de catálogo raíz instalado. El valor predeterminado es la ruta de acceso absoluta del archivo de catálogo raíz instalado (`<carpeta-instalación>\Altova\RaptorXMLServer2024\etc\RootCatalog.xml`).
    Consulte el apartado Catálogos XML [48] para obtener más información.

- user-catalog

  --user-catalog = *ARCHIVO*
  Especifica la ruta de acceso absoluta a un catálogo XML que debe utilizarse junto con el catálogo raíz. Consulte el apartado Catálogos XML [48] para obtener más información.

- enable-globalresources

  --enable-globalresources = `true|false`
  Habilita la función de recursos globales [55]. Valor predeterminado: `false`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- globalresourceconfig [gc]

  --gc | --globalresourceconfig = *VALOR*
  Especifica la configuración activa del recurso global [55] (y habilita los recursos globales) [55].

- globalresourcefile [gr]

  --gr | --globalresourcefile = *ARCHIVO*
  Especifica el archivo de recursos globales [55] (y habilita los recursos globales) [55].

- Opciones comunes
  - error-format

    --error-format = `text|shortxml|longxml`
    Especifica el formato de la salida de error. Los valores posibles son formatos de texto, XML y XML detallado (`longxml`). Valor predeterminado: `text`.

  - error-limit

    --error-limit = *N | unlimited*
    Especifica el límite de errores con un valor comprendido entre `1` y `9999` o `unlimited` (ilimitado). El valor predeterminado es `100`. Cuando se alcanza el límite de error, se detiene la validación. Esta opción es muy práctica a la hora de limitar el uso del procesador durante la validación o transformación.

  - info-limit

    --info-limit = *N | unlimited*
    Indica el límite del mensaje de información dentro del rango `1-65535` or `unlimited`. Si se alcanza el límite de información indicado, el procesamiento continúa pero ya no se informa de más mensajes. El valor predeterminado es `100`.

  - help

    --help
    Muestra el texto de ayuda para el comando. Por ejemplo `valany --h`. (Otra opción es usar el comando `help` con un argumento. Por ejemplo: `help valany`).

© 2017-2023 Altova GmbH

- listfile

  --listfile = `true|false`
  Si el valor es `true`, el argumento *`ArchivoEntrada`* del comando se entiende como un archivo de texto que contiene un nombre de archivo por línea. Otra opción es enumerar los archivos en la ILC, separados por un espacio. No obstante, recuerde que las ILC tienen un límite de caracteres. Además, no olvide que la opción `--listfile` solamente afecta a los argumentos y no a las opciones.
  Valor predeterminado: `false`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- log-output

  --log-output = *`ARCHIVO`*
  Escribe el registro de salida en la URL de archivo indicada. Compruebe que la ILC tiene permiso de escritura en la ubicación de destino.

- network-timeout

  --network-timeout = *`VALOR`*
  Indica el tiempo de espera en milisegundos para operaciones remotas de entrada y salida. Valor predeterminado: `40000`.

- recurse

  --recurse = `true|false`
  Esta opción se utiliza para seleccionar ficheros dentro de subdirectorios (incluso en archivos ZIP). Si el valor es `true`, el argumento *`ArchivoEntrada`* del comando seleccionará el fichero seleccionado también en los subdirectorios. Por ejemplo: `"test.zip|zip\test.xml"` seleccionará los ficheros llamados `test.xml` en todos los subdirectorios de la carpeta ZIP. Si quiere puede usar los caracteres comodín `*` y `?`. Por ejemplo: `*.xml` seleccionaría todos los ficheros de la carpeta ZIP que tengan la extensión `.xml`. Valor predeterminado: `false`
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- verbose

  --verbose = `true|false`
  Si el valor es `true`, se genera información adicional durante la validación. Valor predeterminado es `false`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- verbose-output

  --verbose-output = *`ARCHIVO`*
  Escribe el resultado detallado en el *`ARCHIVO`* indicado.

- version

  --version
  Muestra el número de versión de RaptorXML Server. Si se utiliza con un comando, escriba la opción `--version` antes del comando.

- warning-limit

  --warning-limit = *N | unlimited*
  Especifica el límite de advertencia en el rango `1-65535` o `unlimited` (ilimitado). El procesamiento continua si se alcanza el límite pero no se registrarán más advertencias. El valor por defecto es `100`.

5.5.6 valavroschema (avroschema)

El comando **valavroschema | avroschema** valida documentos de esquema Avro con la especificación Avro schema.

Windows	**RaptorXML valavroschema | avroschema [opciones]** ***EsquemaAvro***
Linux	**raptorxml valavroschema | avroschema [opciones]** ***EsquemaAvro***
Mac	**raptorxml valavroschema | avroschema [opciones]** ***EsquemaAvro***

El argumento *EsquemaAvro* es el documento de esquema Avro que se debe validar.

Si desea validar varios documentos de esquema Avro, tiene dos opciones: (i) enumerar los archivos de esquema que se deben validar en la línea de comandos, separados por espacios o (ii) enumerar los archivos que se deben validar en un archivo de texto (`.txt`) donde aparece un nombre de archivo por línea y dar este archivo de texto como argumento *EsquemaAvro* junto con la opción `--listfile`[238] con valor `true` (*ver lista de opciones más abajo*).

Ejemplos

- **raptorxml** valavroschema c:\MiEsquemaAvro.avsc
- **raptorxml** valavroschema c:\MiEsquemaAvro01.avsc c:\MiEsquemaAvro02.avsc
- **raptorxml** avroschema --listfile=true c:\MiListaArchivos.txt

- Mayúsculas/minúsculas y barras en la línea de comandos

  RaptorXML (y **RaptorXMLServer** para comandos administrativos) *en Windows*
  raptorxml (y **raptorxmlserver** para comandos administrativos) *en Windows y Unix (Linux, Mac)*

  * Las minúsculas (`raptorxml` y `raptorxmlserver`) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que las mayúsculas (`RaptorXML`) funcionan solamente en Windows y Mac.
  * Use la barra diagonal en Linux y Mac y la barra diagonal inversa en Windows.

- Barra diagonal inversa y espacios en sistemas Windows

  En sistemas Windows: si hay espacios o caracteres especiales (por ejemplo en los nombres de archivos o de carpetas, empresas, personas o productos) debe usar comillas: por ejemplo, **"Mi archivo"**. Sin

 © 2017-2023 Altova GmbH

embargo, debe tener en cuenta que una barra diagonal inversa seguida de comillas dobles (por ejemplo, `"C:\Mi Directorio\"`) es posible que no se lea correctamente. Esto se debe a que la barra diagonal inversa también se usa para empezar secuencias de escape y la secuencia de escape `\"` también usa las comillas dobles. Para evitar secuencia de caracteres puede añadir otra barra diagonal inversa: `\\"`. En resumen: si necesita escribir una ruta que contenga espacios y una barra diagonal inversa, esta es la mejor forma de hacerlo: `"C:\Mi Directorio\\"`.

Opciones

Las opciones del comando aparecen más abajo divididas en grupos. Los valores se pueden dar sin comillas excepto en estos dos casos: (i) cuando la cadena de valor contiene espacios y (ii) cuando en la descripción de la opción se indique explícitamente que es necesario el uso de comillas. Si una opción toma un valor booleano y no se indica ningún valor, entonces el valor predeterminado de la opción es `TRUE`. Use la opción **`--h, --help`** para ver información sobre el comando.

- Procesamiento
  - listfile

    `--listfile =` `true|false`
    Si el valor es `true`, el argumento *`ArchivoEntrada`* del comando se entiende como un archivo de texto que contiene un nombre de archivo por línea. Otra opción es enumerar los archivos en la ILC, separados por un espacio. No obstante, recuerde que las ILC tienen un límite de caracteres. Además, no olvide que la opción `--listfile` solamente afecta a los argumentos y no a las opciones.
    Valor predeterminado: `false`.
    Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

  - recurse

    `--recurse =` `true|false`
    Esta opción se utiliza para seleccionar ficheros dentro de subdirectorios (incluso en archivos ZIP). Si el valor es `true`, el argumento *`ArchivoEntrada`* del comando seleccionará el fichero seleccionado también en los subdirectorios. Por ejemplo: `"test.zip|zip\test.xml"` seleccionará los ficheros llamados `test.xml` en todos los subdirectorios de la carpeta ZIP. Si quiere puede usar los caracteres comodín `*` y `?`. Por ejemplo: `*.xml` seleccionaría todos los ficheros de la carpeta ZIP que tengan la extensión `.xml`. Valor predeterminado: `false`
    Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- Catálogos y recursos globales
  - catalog

    `--catalog =` *`ARCHIVO`*
    Especifica la ruta de acceso absoluta a un archivo de catálogo que no está en el archivo de catálogo raíz instalado. El valor predeterminado es la ruta de acceso absoluta del archivo de catálogo raíz instalado (`<carpeta-instalación>\Altova\RaptorXMLServer2024\etc\RootCatalog.xml`).
    Consulte el apartado Catálogos XML [48] para obtener más información.

- user-catalog

  --user-catalog = *ARCHIVO*
  Especifica la ruta de acceso absoluta a un catálogo XML que debe utilizarse junto con el catálogo raíz. Consulte el apartado Catálogos XML[48] para obtener más información.

- enable-globalresources

  --enable-globalresources = `true|false`
  Habilita la función de recursos globales[55]. Valor predeterminado: `false`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- globalresourceconfig [gc]

  --gc | --globalresourceconfig = *VALOR*
  Especifica la configuración activa del recurso global[55] (y habilita los recursos globales)[55].

- globalresourcefile [gr]

  --gr | --globalresourcefile = *ARCHIVO*
  Especifica el archivo de recursos globales[55] (y habilita los recursos globales)[55].

- Opciones comunes
  - error-format

    --error-format = `text|shortxml|longxml`
    Especifica el formato de la salida de error. Los valores posibles son formatos de texto, XML y XML detallado (`longxml`). Valor predeterminado: `text`.

  - error-limit

    --error-limit = *N | unlimited*
    Especifica el límite de errores con un valor comprendido entre `1` y `9999` o `unlimited` (ilimitado). El valor predeterminado es `100`. Cuando se alcanza el límite de error, se detiene la validación. Esta opción es muy práctica a la hora de limitar el uso del procesador durante la validación o transformación.

  - info-limit

    --info-limit = *N | unlimited*
    Indica el límite del mensaje de información dentro del rango `1-65535` or `unlimited`. Si se alcanza el límite de información indicado, el procesamiento continúa pero ya no se informa de más mensajes. El valor predeterminado es `100`.

  - help

    --help
    Muestra el texto de ayuda para el comando. Por ejemplo `valany --h`. (Otra opción es usar el comando `help` con un argumento. Por ejemplo: `help valany`).

  - listfile

© 2017-2023 Altova GmbH

--listfile = true|false
Si el valor es true, el argumento *ArchivoEntrada* del comando se entiende como un archivo de texto que contiene un nombre de archivo por línea. Otra opción es enumerar los archivos en la ILC, separados por un espacio. No obstante, recuerde que las ILC tienen un límite de caracteres. Además, no olvide que la opción --listfile solamente afecta a los argumentos y no a las opciones.
Valor predeterminado: false.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en true.

- log-output

--log-output = *ARCHIVO*
Escribe el registro de salida en la URL de archivo indicada. Compruebe que la ILC tiene permiso de escritura en la ubicación de destino.

- network-timeout

--network-timeout = *VALOR*
Indica el tiempo de espera en milisegundos para operaciones remotas de entrada y salida. Valor predeterminado: 40000.

- recurse

--recurse = true|false
Esta opción se utiliza para seleccionar ficheros dentro de subdirectorios (incluso en archivos ZIP). Si el valor es true, el argumento *ArchivoEntrada* del comando seleccionará el fichero seleccionado también en los subdirectorios. Por ejemplo: "test.zip|zip\test.xml" seleccionará los ficheros llamados test.xml en todos los subdirectorios de la carpeta ZIP. Si quiere puede usar los caracteres comodín * y ?. Por ejemplo: *.xml seleccionaría todos los ficheros de la carpeta ZIP que tengan la extensión .xml. Valor predeterminado: false
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en true.

- verbose

--verbose = true|false
Si el valor es true, se genera información adicional durante la validación. Valor predeterminado es false.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en true.

- verbose-output

--verbose-output = *ARCHIVO*
Escribe el resultado detallado en el *ARCHIVO* indicado.

- version

--version
Muestra el número de versión de RaptorXML Server. Si se utiliza con un comando, escriba la opción --version antes del comando.

▼ warning-limit

--warning-limit = *N | unlimited*
Especifica el límite de advertencia en el rango `1-65535` o `unlimited` (ilimitado). El procesamiento continua si se alcanza el límite pero no se registrarán más advertencias. El valor por defecto es `100`.

5.5.7 valjsonschema (jsonschema)

El comando **valjsonschema | jsonschema** valida documentos de esquema JSON conforme a distintas especificaciones del JSON Schema (se configura con la opción **jsonschema-version**).

Windows	**RaptorXML valjsonschema | jsonschema [opciones]** ***ArchivoEntrada***
Linux	**raptorxml valjsonschema | jsonschema [opciones]** ***ArchivoEntrada***
Mac	**raptorxml valjsonschema | jsonschema [opciones]** ***ArchivoEntrada***

- El argumento *ArchivoEntrada* es el documento de esquema JSON que se debe validar.
- Si desea validar varios documentos, tiene dos opciones: (i) enumerar los archivos que se deben validar en la línea de comandos, separados por espacios o (ii) enumerar los archivos que se deben validar en un archivo de texto (.txt) donde aparece un nombre de archivo por línea y dar este archivo de texto como argumento *ArchivoEntrada* junto con la opción --listfile[238] con valor true (ver lista de opciones más abajo).

Ejemplos

Ejemplos del comando **valjsonschema**:

- **raptorxml** valjsonschema c:\MiEsquemaJSON.json
- **raptorxml** jsonschema c:\MiEsquemaJSON-01.json c:\MiEsquemaJSON-02.json
- **raptorxml** jsonschema --listfile=true c:\ListaArchivos.txt

▼ Mayúsculas/minúsculas y barras en la línea de comandos

RaptorXML (y **RaptorXMLServer** para comandos administrativos) *en Windows*
raptorxml (y **raptorxmlserver** para comandos administrativos) *en Windows y Unix (Linux, Mac)*

* Las minúsculas (`raptorxml` y `raptorxmlserver`) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que las mayúsculas (`RaptorXML`) funcionan solamente en Windows y Mac.
* Use la barra diagonal en Linux y Mac y la barra diagonal inversa en Windows.

▼ Barra diagonal inversa y espacios en sistemas Windows

En sistemas Windows: si hay espacios o caracteres especiales (por ejemplo en los nombres de archivos o de carpetas, empresas, personas o productos) debe usar comillas: por ejemplo, **"Mi archivo"**. Sin embargo, debe tener en cuenta que una barra diagonal inversa seguida de comillas dobles (por ejemplo, **"C:\Mi Directorio\"**) es posible que no se lea correctamente. Esto se debe a que la barra diagonal

 © 2017-2023 Altova GmbH

inversa también se usa para empezar secuencias de escape y la secuencia de escape `\"` también usa las comillas dobles. Para evitar secuencia de caracteres puede añadir otra barra diagonal inversa: `\\"`. En resumen: si necesita escribir una ruta que contenga espacios y una barra diagonal inversa, esta es la mejor forma de hacerlo: `"C:\Mi Directorio\\"`.

Opciones

Las opciones del comando aparecen más abajo divididas en grupos. Los valores se pueden dar sin comillas excepto en estos dos casos: (i) cuando la cadena de valor contiene espacios y (ii) cuando en la descripción de la opción se indique explícitamente que es necesario el uso de comillas. Si una opción toma un valor booleano y no se indica ningún valor, entonces el valor predeterminado de la opción es `TRUE`. Use la opción **`--h, --help`** para ver información sobre el comando.

- Validación y procesamiento
  - listfile

    `--listfile =` `true|false`
    Si el valor es `true`, el argumento *`ArchivoEntrada`* del comando se entiende como un archivo de texto que contiene un nombre de archivo por línea. Otra opción es enumerar los archivos en la ILC, separados por un espacio. No obstante, recuerde que las ILC tienen un límite de caracteres. Además, no olvide que la opción `--listfile` solamente afecta a los argumentos y no a las opciones.
    Valor predeterminado: `false`.
    Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

  - recurse

    `--recurse =` `true|false`
    Esta opción se utiliza para seleccionar ficheros dentro de subdirectorios (incluso en archivos ZIP). Si el valor es `true`, el argumento *`ArchivoEntrada`* del comando seleccionará el fichero seleccionado también en los subdirectorios. Por ejemplo: `"test.zip|zip\test.xml"` seleccionará los ficheros llamados `test.xml` en todos los subdirectorios de la carpeta ZIP. Si quiere puede usar los caracteres comodín `*` y `?`. Por ejemplo: `*.xml` seleccionaría todos los ficheros de la carpeta ZIP que tengan la extensión `.xml`. Valor predeterminado: `false`
    Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- Opciones de validación JSON
  - additional-schema

    `--additional-schema =` *`FILE`*
    Indica los URIs de un documento de esquema suplementario. El esquema suplementario se carga desde el principal y se puede hacer referencia a él también desde el esquema principal con la propiedad **`id`** o **`$id`** del esquema suplementario.

  - disable-format-checks

--disable-format-checks = `true|false`
Deshabilita la validación semántica impuesta por el atributo de formato. El valor predeterminado es `false`.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- jsonschema-version

  --jsonschema-version = `draft04|draft06|draft07|2019-09|2020-12|latest|detect`
  Indica qué versión de la versión del borrador de la especificación JSON Schema usar. El valor predeterminado es `detect`.

- strict-integer-checks

  --strict-integer-checks = `true|false`
  Indica si se debe usar la comprobación estricta de números enteros del borrador draft-04 con esquemas posteriores donde las comprobaciones de números enteros son más laxas. Por ejemplo, `1.0` no es un número entero válido en draft-04 pero sí en borradores posteriores. Esta opción no afecta a los esquemas de draft-04. El valor predeterminado de esta opción es `false`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- Catálogos y recursos globales

  - catalog

    --catalog = *ARCHIVO*
    Especifica la ruta de acceso absoluta a un archivo de catálogo que no está en el archivo de catálogo raíz instalado. El valor predeterminado es la ruta de acceso absoluta del archivo de catálogo raíz instalado (`<carpeta-instalación>\Altova\RaptorXMLServer2024\etc\RootCatalog.xml`). Consulte el apartado Catálogos XML [48] para obtener más información.

  - user-catalog

    --user-catalog = *ARCHIVO*
    Especifica la ruta de acceso absoluta a un catálogo XML que debe utilizarse junto con el catálogo raíz. Consulte el apartado Catálogos XML [48] para obtener más información.

  - enable-globalresources

    --enable-globalresources = `true|false`
    Habilita la función de recursos globales [55]. Valor predeterminado: `false`.
    Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

  - globalresourceconfig [gc]

    --gc | --globalresourceconfig = *VALOR*
    Especifica la configuración activa del recurso global [55] (y habilita los recursos globales) [55].

  - globalresourcefile [gr]

    --gr | --globalresourcefile = *ARCHIVO*

© 2017-2023 Altova GmbH

Especifica el archivo de recursos globales[55] (y habilita los recursos globales)[55].

- Opciones comunes
  - error-format

    --error-format = `text|shortxml|longxml`
    Especifica el formato de la salida de error. Los valores posibles son formatos de texto, XML y XML detallado (`longxml`). Valor predeterminado: `text`.

  - error-limit

    --error-limit = *N | unlimited*
    Especifica el límite de errores con un valor comprendido entre `1` y `9999` o `unlimited` (ilimitado). El valor predeterminado es `100`. Cuando se alcanza el límite de error, se detiene la validación. Esta opción es muy práctica a la hora de limitar el uso del procesador durante la validación o transformación.

  - info-limit

    --info-limit = *N | unlimited*
    Indica el límite del mensaje de información dentro del rango `1-65535` or `unlimited`. Si se alcanza el límite de información indicado, el procesamiento continúa pero ya no se informa de más mensajes. El valor predeterminado es `100`.

  - help

    --help
    Muestra el texto de ayuda para el comando. Por ejemplo `valany --h`. (Otra opción es usar el comando `help` con un argumento. Por ejemplo: `help valany`).

  - listfile

    --listfile = `true|false`
    Si el valor es `true`, el argumento *ArchivoEntrada* del comando se entiende como un archivo de texto que contiene un nombre de archivo por línea. Otra opción es enumerar los archivos en la ILC, separados por un espacio. No obstante, recuerde que las ILC tienen un límite de caracteres. Además, no olvide que la opción `--listfile` solamente afecta a los argumentos y no a las opciones.
    Valor predeterminado: `false`.
    Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

  - log-output

    --log-output = *ARCHIVO*
    Escribe el registro de salida en la URL de archivo indicada. Compruebe que la ILC tiene permiso de escritura en la ubicación de destino.

  - network-timeout

    --network-timeout = *VALOR*
    Indica el tiempo de espera en milisegundos para operaciones remotas de entrada y salida. Valor

predeterminado: `40000`.

- recurse

  `--recurse =` `true|false`
  Esta opción se utiliza para seleccionar ficheros dentro de subdirectorios (incluso en archivos ZIP). Si el valor es `true`, el argumento *`ArchivoEntrada`* del comando seleccionará el fichero seleccionado también en los subdirectorios. Por ejemplo: `"test.zip|zip\test.xml"` seleccionará los ficheros llamados `test.xml` en todos los subdirectorios de la carpeta ZIP. Si quiere puede usar los caracteres comodín `*` y `?`. Por ejemplo: `*.xml` seleccionaría todos los ficheros de la carpeta ZIP que tengan la extensión `.xml`. Valor predeterminado: `false`
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- verbose

  `--verbose =` `true|false`
  Si el valor es `true`, se genera información adicional durante la validación. Valor predeterminado es `false`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- verbose-output

  `--verbose-output =` *`ARCHIVO`*
  Escribe el resultado detallado en el *`ARCHIVO`* indicado.

- version

  `--version`
  Muestra el número de versión de RaptorXML Server. Si se utiliza con un comando, escriba la opción `--version` antes del comando.

- warning-limit

  `--warning-limit =` *`N | unlimited`*
  Especifica el límite de advertencia en el rango `1-65535` o `unlimited` (ilimitado). El procesamiento continua si se alcanza el límite pero no se registrarán más advertencias. El valor por defecto es `100`.

5.5.8 valjson (json)

El comando **`valjson | json`** valida documentos de instancia JSON con el esquema JSON dado con la opción `--schema`.

Windows	**`RaptorXML valjson	json [opciones] --jsonschema=`*`Archivo ArchivoEntrada`*
Linux	**`raptorxml valjson	json [opciones] --jsonschema=`*`Archivo ArchivoEntrada`*

© 2017-2023 Altova GmbH

Mac **raptorxml valjson | json [opciones] --jsonschema=*Archivo* *ArchivoEntrada***

El argumento *ArchivoEntrada* es el documento de instancia que se debe validar.

Si desea validar varios documentos, tiene dos opciones: (i) enumerar los archivos que se deben validar en la línea de comandos, separados por espacios o (ii) enumerar los archivos que se deben validar en un archivo de texto (.txt) donde aparece un nombre de archivo por línea y dar este archivo de texto como argumento *ArchivoEntrada* junto con la opción --listfile[238] con valor true (*ver lista de opciones más abajo*).

Ejemplos

- **raptorxml** valjson --jsonschema=c:\MiEsquemaJSON.json c:\MiInstanciaJSON.json
- **raptorxml** json --jsonschema=c:\MiEsquemaJSON.json c:\MiInstanciaJSON-01.json c:\MiInstanciaJSON-02.json
- **raptorxml** json --jsonschema=c:\MiEsquemaJSON.json --listfile=true c:\ListaArchivos.txt

▼ Mayúsculas/minúsculas y barras en la línea de comandos

RaptorXML (y **RaptorXMLServer** para comandos administrativos) *en Windows*
raptorxml (y **raptorxmlserver** para comandos administrativos) *en Windows y Unix (Linux, Mac)*

* Las minúsculas (raptorxml y raptorxmlserver) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que las mayúsculas (RaptorXML) funcionan solamente en Windows y Mac.
* Use la barra diagonal en Linux y Mac y la barra diagonal inversa en Windows.

▼ Barra diagonal inversa y espacios en sistemas Windows

En sistemas Windows: si hay espacios o caracteres especiales (por ejemplo en los nombres de archivos o de carpetas, empresas, personas o productos) debe usar comillas: por ejemplo, **"Mi archivo"**. Sin embargo, debe tener en cuenta que una barra diagonal inversa seguida de comillas dobles (por ejemplo, **"C:\Mi Directorio\"**) es posible que no se lea correctamente. Esto se debe a que la barra diagonal inversa también se usa para empezar secuencias de escape y la secuencia de escape **\"** también usa las comillas dobles. Para evitar secuencia de caracteres puede añadir otra barra diagonal inversa: **\\"**. En resumen: si necesita escribir una ruta que contenga espacios y una barra diagonal inversa, esta es la mejor forma de hacerlo: **"C:\Mi Directorio\\"**.

Opciones

Las opciones del comando aparecen más abajo divididas en grupos. Los valores se pueden dar sin comillas excepto en estos dos casos: (i) cuando la cadena de valor contiene espacios y (ii) cuando en la descripción de la opción se indique explícitamente que es necesario el uso de comillas. Si una opción toma un valor booleano y no se indica ningún valor, entonces el valor predeterminado de la opción es TRUE. Use la opción **--h, --help** para ver información sobre el comando.

▼ Validación y procesamiento

▼ schema**, jsonschema**

--schema = *ARCHIVO*, **--jsonschema =** *ARCHIVO*
Especifica el documento del esquema JSON que se va usar para la validación de documentos de instancia JSON.

- listfile

  --listfile = `true|false`
  Si el valor es `true`, el argumento *ArchivoEntrada* del comando se entiende como un archivo de texto que contiene un nombre de archivo por línea. Otra opción es enumerar los archivos en la ILC, separados por un espacio. No obstante, recuerde que las ILC tienen un límite de caracteres. Además, no olvide que la opción `--listfile` solamente afecta a los argumentos y no a las opciones.
  Valor predeterminado: `false`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- additional-schema

  --additional-schema = *FILE*
  Indica los URIs de un documento de esquema suplementario. El esquema suplementario se carga desde el principal y se puede hacer referencia a él también desde el esquema principal con la propiedad **id** o **$id** del esquema suplementario.

- recurse

  --recurse = `true|false`
  Esta opción se utiliza para seleccionar ficheros dentro de subdirectorios (incluso en archivos ZIP). Si el valor es `true`, el argumento *ArchivoEntrada* del comando seleccionará el fichero seleccionado también en los subdirectorios. Por ejemplo: `"test.zip|zip\test.xml"` seleccionará los ficheros llamados `test.xml` en todos los subdirectorios de la carpeta ZIP. Si quiere puede usar los caracteres comodín `*` y `?`. Por ejemplo: `*.xml` seleccionaría todos los ficheros de la carpeta ZIP que tengan la extensión `.xml`. Valor predeterminado: `false`
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- json5

  --json5 = `true|false`
  Habilita la compatibilidad con JSON5. Valor predeterminado: `false`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- jsonc

  --jsonc = `true|false`
  Habilita la compatibilidad con los comentarios en JSON. El valor predeterminado es `false`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- json-lines

  --json-lines = `true|false`
  Habilita la compatibilidad con Líneas JSON (es decir, un valor JSON por línea). El valor

 © 2017-2023 Altova GmbH

predeterminado es `false`.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- jsonschema-version

  `--jsonschema-version =` `draft04|draft06|draft07|2019-09|2020-12|latest|detect`
  Indica qué versión de la versión del borrador de la especificación JSON Schema usar. El valor predeterminado es `detect`.

- disable-format-checks

  `--disable-format-checks =` `true|false`
  Deshabilita la validación semántica impuesta por el atributo de formato. El valor predeterminado es `false`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- strict-integer-checks

  `--strict-integer-checks =` `true|false`
  Indica si se debe usar la comprobación estricta de números enteros del borrador draft-04 con esquemas posteriores donde las comprobaciones de números enteros son más laxas. Por ejemplo, `1.0` no es un número entero válido en draft-04 pero sí en borradores posteriores. Esta opción no afecta a los esquemas de draft-04. El valor predeterminado de esta opción es `false`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- Catálogos y recursos globales
  - catalog

    `--catalog =` *`ARCHIVO`*
    Especifica la ruta de acceso absoluta a un archivo de catálogo que no está en el archivo de catálogo raíz instalado. El valor predeterminado es la ruta de acceso absoluta del archivo de catálogo raíz instalado (`<carpeta-instalación>\Altova\RaptorXMLServer2024\etc\RootCatalog.xml`). Consulte el apartado Catálogos XML [48] para obtener más información.

  - user-catalog

    `--user-catalog =` *`ARCHIVO`*
    Especifica la ruta de acceso absoluta a un catálogo XML que debe utilizarse junto con el catálogo raíz. Consulte el apartado Catálogos XML [48] para obtener más información.

  - enable-globalresources

    `--enable-globalresources =` `true|false`
    Habilita la función de recursos globales [55]. Valor predeterminado: `false`.
    Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

  - globalresourceconfig [gc]

    `--gc | --globalresourceconfig =` *`VALOR`*

Especifica la configuración activa del recurso global[55] (y habilita los recursos globales)[55].

- globalresourcefile [gr]

  --gr | --globalresourcefile = *ARCHIVO*
  Especifica el archivo de recursos globales[55] (y habilita los recursos globales)[55].

- Opciones comunes
  - error-format

    --error-format = `text|shortxml|longxml`
    Especifica el formato de la salida de error. Los valores posibles son formatos de texto, XML y XML detallado (`longxml`). Valor predeterminado: `text`.

  - error-limit

    --error-limit = *N | unlimited*
    Especifica el límite de errores con un valor comprendido entre `1` y `9999` o `unlimited` (ilimitado). El valor predeterminado es `100`. Cuando se alcanza el límite de error, se detiene la validación. Esta opción es muy práctica a la hora de limitar el uso del procesador durante la validación o transformación.

  - info-limit

    --info-limit = *N | unlimited*
    Indica el límite del mensaje de información dentro del rango `1-65535` or `unlimited`. Si se alcanza el límite de información indicado, el procesamiento continúa pero ya no se informa de más mensajes. El valor predeterminado es `100`.

  - help

    --help
    Muestra el texto de ayuda para el comando. Por ejemplo `valany --h`. (Otra opción es usar el comando `help` con un argumento. Por ejemplo: `help valany`).

  - listfile

    --listfile = `true|false`
    Si el valor es `true`, el argumento *ArchivoEntrada* del comando se entiende como un archivo de texto que contiene un nombre de archivo por línea. Otra opción es enumerar los archivos en la ILC, separados por un espacio. No obstante, recuerde que las ILC tienen un límite de caracteres. Además, no olvide que la opción `--listfile` solamente afecta a los argumentos y no a las opciones.
    Valor predeterminado: `false`.
    Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

  - log-output

    --log-output = *ARCHIVO*
    Escribe el registro de salida en la URL de archivo indicada. Compruebe que la ILC tiene permiso de escritura en la ubicación de destino.

 © 2017-2023 Altova GmbH

- network-timeout

  --network-timeout = *VALOR*
  Indica el tiempo de espera en milisegundos para operaciones remotas de entrada y salida. Valor predeterminado: `40000`.

- recurse

  --recurse = `true|false`
  Esta opción se utiliza para seleccionar ficheros dentro de subdirectorios (incluso en archivos ZIP). Si el valor es `true`, el argumento *ArchivoEntrada* del comando seleccionará el fichero seleccionado también en los subdirectorios. Por ejemplo: `"test.zip|zip\test.xml"` seleccionará los ficheros llamados `test.xml` en todos los subdirectorios de la carpeta ZIP. Si quiere puede usar los caracteres comodín `*` y `?`. Por ejemplo: `*.xml` seleccionaría todos los ficheros de la carpeta ZIP que tengan la extensión `.xml`. Valor predeterminado: `false`
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- verbose

  --verbose = `true|false`
  Si el valor es `true`, se genera información adicional durante la validación. Valor predeterminado es `false`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- verbose-output

  --verbose-output = *ARCHIVO*
  Escribe el resultado detallado en el *ARCHIVO* indicado.

- version

  --version
  Muestra el número de versión de RaptorXML Server. Si se utiliza con un comando, escriba la opción `--version` antes del comando.

- warning-limit

  --warning-limit = *N | unlimited*
  Especifica el límite de advertencia en el rango `1-65535` o `unlimited` (ilimitado). El procesamiento continua si se alcanza el límite pero no se registrarán más advertencias. El valor por defecto es `100`.

5.5.9 wfjson

El comando **wfjson** comprueba si el formato de los documentos JSON se ajusta a las normas de la especificación ECMA-404.

Windows	**RaptorXML wfjson [opciones] *ArchivoEntrada***
Linux	**raptorxml wfjson [opciones] *ArchivoEntrada***
Mac	**raptorxml wfjson [opciones] *ArchivoEntrada***

El argumento *ArchivoEntrada* es el documento JSON (esquema o instancia) cuyo formato debe revisarse.

Si desea revisar el formato de varios documentos, tiene dos opciones: (i) enumerar los archivos que se deben revisar en la línea de comandos, separados por espacios o (ii) enumerar los archivos que se deben revisar en un archivo de texto (`.txt`) donde aparece un nombre de archivo por línea y dar este archivo de texto como argumento *ArchivoEntrada* junto con la opción `--listfile`[238] con valor `true` (*ver lista de opciones más abajo*).

Ejemplos

- **raptorxml** wfjson c:\MiArchivoJSON.json
- **raptorxml** wfjson c:\MiArchivoJSON-01.json c:\MiArchivoJSON-02.json
- **raptorxml** wfjson --listfile=true c:\ListaArchivos.txt

▼ Mayúsculas/minúsculas y barras en la línea de comandos

RaptorXML (y **RaptorXMLServer** para comandos administrativos) *en Windows*
raptorxml (y **raptorxmlserver** para comandos administrativos) *en Windows y Unix (Linux, Mac)*

* Las minúsculas (`raptorxml` y `raptorxmlserver`) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que las mayúsculas (`RaptorXML`) funcionan solamente en Windows y Mac.
* Use la barra diagonal en Linux y Mac y la barra diagonal inversa en Windows.

▼ Barra diagonal inversa y espacios en sistemas Windows

En sistemas Windows: si hay espacios o caracteres especiales (por ejemplo en los nombres de archivos o de carpetas, empresas, personas o productos) debe usar comillas: por ejemplo, **"Mi archivo"**. Sin embargo, debe tener en cuenta que una barra diagonal inversa seguida de comillas dobles (por ejemplo, **"C:\Mi Directorio\"**) es posible que no se lea correctamente. Esto se debe a que la barra diagonal inversa también se usa para empezar secuencias de escape y la secuencia de escape **\"** también usa las comillas dobles. Para evitar secuencia de caracteres puede añadir otra barra diagonal inversa: **\\"**. En resumen: si necesita escribir una ruta que contenga espacios y una barra diagonal inversa, esta es la mejor forma de hacerlo: **"C:\Mi Directorio\\"**.

Opciones

Las opciones del comando aparecen más abajo divididas en grupos. Los valores se pueden dar sin comillas excepto en estos dos casos: (i) cuando la cadena de valor contiene espacios y (ii) cuando en la descripción de la opción se indique explícitamente que es necesario el uso de comillas. Si una opción toma un valor booleano y no se indica ningún valor, entonces el valor predeterminado de la opción es `TRUE`. Use la opción **--h, --help** para ver información sobre el comando.

▼ Validación y procesamiento

 © 2017-2023 Altova GmbH

- json5

  --json5 = `true|false`
  Habilita la compatibilidad con JSON5. Valor predeterminado: `false`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- jsonc

  --jsonc = `true|false`
  Habilita la compatibilidad con los comentarios en JSON. El valor predeterminado es `false`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- json-lines

  --json-lines = `true|false`
  Habilita la compatibilidad con Líneas JSON (es decir, un valor JSON por línea). El valor predeterminado es `false`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- listfile

  --listfile = `true|false`
  Si el valor es `true`, el argumento *`ArchivoEntrada`* del comando se entiende como un archivo de texto que contiene un nombre de archivo por línea. Otra opción es enumerar los archivos en la ILC, separados por un espacio. No obstante, recuerde que las ILC tienen un límite de caracteres. Además, no olvide que la opción `--listfile` solamente afecta a los argumentos y no a las opciones.
  Valor predeterminado: `false`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- recurse

  --recurse = `true|false`
  Esta opción se utiliza para seleccionar ficheros dentro de subdirectorios (incluso en archivos ZIP). Si el valor es `true`, el argumento *`ArchivoEntrada`* del comando seleccionará el fichero seleccionado también en los subdirectorios. Por ejemplo: `"test.zip|zip\test.xml"` seleccionará los ficheros llamados `test.xml` en todos los subdirectorios de la carpeta ZIP. Si quiere puede usar los caracteres comodín `*` y `?`. Por ejemplo: `*.xml` seleccionaría todos los ficheros de la carpeta ZIP que tengan la extensión `.xml`. Valor predeterminado: `false`
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- Catálogos y recursos globales

  - catalog

    --catalog = *`ARCHIVO`*
    Especifica la ruta de acceso absoluta a un archivo de catálogo que no está en el archivo de catálogo raíz instalado. El valor predeterminado es la ruta de acceso absoluta del archivo de catálogo raíz

instalado (`<carpeta-instalación>\Altova\RaptorXMLServer2024\etc\RootCatalog.xml`). Consulte el apartado Catálogos XML [48] para obtener más información.

- user-catalog

  `--user-catalog =` *ARCHIVO*
  Especifica la ruta de acceso absoluta a un catálogo XML que debe utilizarse junto con el catálogo raíz. Consulte el apartado Catálogos XML [48] para obtener más información.

- enable-globalresources

  `--enable-globalresources =` `true|false`
  Habilita la función de recursos globales [55]. Valor predeterminado: `false`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- globalresourceconfig [gc]

  `--gc | --globalresourceconfig =` *VALOR*
  Especifica la configuración activa del recurso global [55] (y habilita los recursos globales) [55].

- globalresourcefile [gr]

  `--gr | --globalresourcefile =` *ARCHIVO*
  Especifica el archivo de recursos globales [55] (y habilita los recursos globales) [55].

- Opciones comunes
  - error-format

    `--error-format =` `text|shortxml|longxml`
    Especifica el formato de la salida de error. Los valores posibles son formatos de texto, XML y XML detallado (`longxml`). Valor predeterminado: `text`.

  - error-limit

    `--error-limit =` *N | unlimited*
    Especifica el límite de errores con un valor comprendido entre `1` y `9999` o `unlimited` (ilimitado). El valor predeterminado es `100`. Cuando se alcanza el límite de error, se detiene la validación. Esta opción es muy práctica a la hora de limitar el uso del procesador durante la validación o transformación.

  - info-limit

    `--info-limit =` *N | unlimited*
    Indica el límite del mensaje de información dentro del rango `1-65535` or `unlimited`. Si se alcanza el límite de información indicado, el procesamiento continúa pero ya no se informa de más mensajes. El valor predeterminado es `100`.

  - help

    `--help`
    Muestra el texto de ayuda para el comando. Por ejemplo `valany --h`. (Otra opción es usar el

 © 2017-2023 Altova GmbH

comando `help` con un argumento. Por ejemplo: `help valany`).

- listfile

  `--listfile =` `true|false`
  Si el valor es `true`, el argumento *`ArchivoEntrada`* del comando se entiende como un archivo de texto que contiene un nombre de archivo por línea. Otra opción es enumerar los archivos en la ILC, separados por un espacio. No obstante, recuerde que las ILC tienen un límite de caracteres. Además, no olvide que la opción `--listfile` solamente afecta a los argumentos y no a las opciones.
  Valor predeterminado: `false`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- log-output

  `--log-output =` *`ARCHIVO`*
  Escribe el registro de salida en la URL de archivo indicada. Compruebe que la ILC tiene permiso de escritura en la ubicación de destino.

- network-timeout

  `--network-timeout =` *`VALOR`*
  Indica el tiempo de espera en milisegundos para operaciones remotas de entrada y salida. Valor predeterminado: `40000`.

- recurse

  `--recurse =` `true|false`
  Esta opción se utiliza para seleccionar ficheros dentro de subdirectorios (incluso en archivos ZIP). Si el valor es `true`, el argumento *`ArchivoEntrada`* del comando seleccionará el fichero seleccionado también en los subdirectorios. Por ejemplo: `"test.zip|zip\test.xml"` seleccionará los ficheros llamados `test.xml` en todos los subdirectorios de la carpeta ZIP. Si quiere puede usar los caracteres comodín `*` y `?`. Por ejemplo: `*.xml` seleccionaría todos los ficheros de la carpeta ZIP que tengan la extensión `.xml`. Valor predeterminado: `false`
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- verbose

  `--verbose =` `true|false`
  Si el valor es `true`, se genera información adicional durante la validación. Valor predeterminado es `false`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- verbose-output

  `--verbose-output =` *`ARCHIVO`*
  Escribe el resultado detallado en el *`ARCHIVO`* indicado.

- version

--version
Muestra el número de versión de RaptorXML Server. Si se utiliza con un comando, escriba la opción `--version` antes del comando.

- warning-limit

  --warning-limit = *N | unlimited*
  Especifica el límite de advertencia en el rango `1-65535` o `unlimited` (ilimitado). El procesamiento continua si se alcanza el límite pero no se registrarán más advertencias. El valor por defecto es `100`.

5.5.10 xml2json

El comando **xml2json** convierte un documento de instancia XML en un documento de instancia JSON.

```
raptorxml XML2json [opciones] ArchivoXML
```

- El argumento *ArchivoXML* es el documento XML que se quiere convertir.
- Utilice la opción `--conversion-output` para especificar dónde se debe guardar el archivo JSON generado.

Ejemplo

Ejemplo del comando **xml2json**:

- **raptorxml** xml2json --conversion-output=c:\MisDatosJSON.json c:\MisDatosXML.xml

- Mayúsculas/minúsculas y barras en la línea de comandos

  RaptorXML (y **RaptorXMLServer** para comandos administrativos) *en Windows*
  raptorxml (y **raptorxmlserver** para comandos administrativos) *en Windows y Unix (Linux, Mac)*

  * Las minúsculas (`raptorxml` y `raptorxmlserver`) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que las mayúsculas (`RaptorXML`) funcionan solamente en Windows y Mac.
  * Use la barra diagonal en Linux y Mac y la barra diagonal inversa en Windows.

- Barra diagonal inversa y espacios en sistemas Windows

  En sistemas Windows: si hay espacios o caracteres especiales (por ejemplo en los nombres de archivos o de carpetas, empresas, personas o productos) debe usar comillas: por ejemplo, **"Mi archivo"**. Sin embargo, debe tener en cuenta que una barra diagonal inversa seguida de comillas dobles (por ejemplo, **"C:\Mi Directorio\"**) es posible que no se lea correctamente. Esto se debe a que la barra diagonal inversa también se usa para empezar secuencias de escape y la secuencia de escape **\"** también usa las comillas dobles. Para evitar secuencia de caracteres puede añadir otra barra diagonal inversa: **\\"**. En resumen: si necesita escribir una ruta que contenga espacios y una barra diagonal inversa, esta es la mejor forma de hacerlo: **"C:\Mi Directorio\\"**.

 © 2017-2023 Altova GmbH

Opciones

Las opciones del comando aparecen más abajo divididas en grupos. Los valores se pueden dar sin comillas excepto en estos dos casos: (i) cuando la cadena de valor contiene espacios y (ii) cuando en la descripción de la opción se indique explícitamente que es necesario el uso de comillas. Si una opción toma un valor booleano y no se indica ningún valor, entonces el valor predeterminado de la opción es `TRUE`. Use la opción **`--h, --help`** para ver información sobre el comando.

Opciones de conversión de datos XML en JSON

Estas opciones definen cómo se gestionan los detalles específicos relacionados con la conversión de datos XML en JSON.

attributes

`--attributes =` `true|false`

Si está establecido en **`true`**, se activa la conversión entre atributos XML y propiedades JSON prefijadas con @. Si no está activada esta opción, no se convertirán ni atributos XML ni propiedades JSON con el prefijo @. El valor predeterminado para esta opción es **`true`**.

Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

comments

`--comments =` `true|false`

Si está establecido en **`true`**, se activa la conversión entre comentarios XML y propiedades JSON prefijadas con #. Si no está activada esta opción, no se convertirán ni atributos XML ni propiedades JSON con el prefijo #. El valor predeterminado para esta opción es **`true`**.

Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

conversion-output, o

`--o, --conversion-output =` `FILE`

Establece el nombre y la ruta de acceso del archivo al que se envía el resultado de la conversión.

ignore-pis

`--ignore-pis =` `true|false`

Si está configurado en **`true`**, se omiten las instrucciones de procesamiento en el documento XML de origen. El valor predeterminado para esta opción es **`false`**.

Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

merge-elements

`--merge-elements =` `true|false`

Si está establecido en **`true`**, se crea una matriz en el documento JSON generado a partir de elementos del mismo nombre y del mismo nivel en el documento XML. El valor predeterminado para esta opción es **`true`**.

Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

merge-text

--merge-text = `true|false`
Si está establecido en **true**, se crea una matriz en el documento JSON generado a partir de nodos de texto del mismo nivel en el documento XML. El valor predeterminado para esta opción es **true**.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- pi

--pi = `true|false`
Si está establecido en **true**, se activa la conversión entre instrucciones de procesamiento XML y propiedades JSON prefijadas con ?. Si no está activada esta opción, no se convertirán ni atributos XML ni propiedades JSON con el prefijo ?. El valor predeterminado para esta opción es **true**.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- pretty-print

--pp, --pretty-print = `true|false`
Si está establecido en **true**, se genera un pretty-print del documento de salida generado. El valor predeterminado para esta opción es **false**.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- text

--text = `true|false`
Si está establecido en **true**, se activa la conversión entre contenido de texto XML y propiedades JSON prefijadas con $. Si no está activada esta opción, no se convertirán ni atributos XML ni propiedades JSON con el prefijo $. El valor predeterminado para esta opción es **true**.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- Opciones comunes
  - error-format

--error-format = `text|shortxml|longxml`
Especifica el formato de la salida de error. Los valores posibles son formatos de texto, XML y XML detallado (`longxml`). Valor predeterminado: `text`.

  - error-limit

--error-limit = *N | unlimited*
Especifica el límite de errores con un valor comprendido entre `1` y `9999` o `unlimited` (ilimitado). El valor predeterminado es `100`. Cuando se alcanza el límite de error, se detiene la validación. Esta opción es muy práctica a la hora de limitar el uso del procesador durante la validación o transformación.

  - info-limit

--info-limit = *N | unlimited*
Indica el límite del mensaje de información dentro del rango `1-65535` or `unlimited`. Si se alcanza el

 © 2017-2023 Altova GmbH

límite de información indicado, el procesamiento continúa pero ya no se informa de más mensajes. El valor predeterminado es `100`.

- help

  `--help`
  Muestra el texto de ayuda para el comando. Por ejemplo `valany --h`. (Otra opción es usar el comando `help` con un argumento. Por ejemplo: `help valany`).

- listfile

  `--listfile =` `true|false`
  Si el valor es `true`, el argumento *`ArchivoEntrada`* del comando se entiende como un archivo de texto que contiene un nombre de archivo por línea. Otra opción es enumerar los archivos en la ILC, separados por un espacio. No obstante, recuerde que las ILC tienen un límite de caracteres. Además, no olvide que la opción `--listfile` solamente afecta a los argumentos y no a las opciones.
  Valor predeterminado: `false`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- log-output

  `--log-output =` *`ARCHIVO`*
  Escribe el registro de salida en la URL de archivo indicada. Compruebe que la ILC tiene permiso de escritura en la ubicación de destino.

- network-timeout

  `--network-timeout =` *`VALOR`*
  Indica el tiempo de espera en milisegundos para operaciones remotas de entrada y salida. Valor predeterminado: `40000`.

- recurse

  `--recurse =` `true|false`
  Esta opción se utiliza para seleccionar ficheros dentro de subdirectorios (incluso en archivos ZIP). Si el valor es `true`, el argumento *`ArchivoEntrada`* del comando seleccionará el fichero seleccionado también en los subdirectorios. Por ejemplo: `"test.zip|zip\test.xml"` seleccionará los ficheros llamados `test.xml` en todos los subdirectorios de la carpeta ZIP. Si quiere puede usar los caracteres comodín `*` y `?`. Por ejemplo: `*.xml` seleccionaría todos los ficheros de la carpeta ZIP que tengan la extensión `.xml`. Valor predeterminado: `false`
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- verbose

  `--verbose =` `true|false`
  Si el valor es `true`, se genera información adicional durante la validación. Valor predeterminado es `false`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- verbose-output

  --verbose-output = *ARCHIVO*
  Escribe el resultado detallado en el *ARCHIVO* indicado.

- version

  --version
  Muestra el número de versión de RaptorXML Server. Si se utiliza con un comando, escriba la opción `--version` antes del comando.

- warning-limit

  --warning-limit = *N | unlimited*
  Especifica el límite de advertencia en el rango `1-65535` o `unlimited` (ilimitado). El procesamiento continua si se alcanza el límite pero no se registrarán más advertencias. El valor por defecto es `100`.

- Catálogos y recursos globales
  - catalog

    --catalog = *ARCHIVO*
    Especifica la ruta de acceso absoluta a un archivo de catálogo que no está en el archivo de catálogo raíz instalado. El valor predeterminado es la ruta de acceso absoluta del archivo de catálogo raíz instalado (`<carpeta-instalación>\Altova\RaptorXMLServer2024\etc\RootCatalog.xml`). Consulte el apartado Catálogos XML [48] para obtener más información.

  - user-catalog

    --user-catalog = *ARCHIVO*
    Especifica la ruta de acceso absoluta a un catálogo XML que debe utilizarse junto con el catálogo raíz. Consulte el apartado Catálogos XML [48] para obtener más información.

  - enable-globalresources

    --enable-globalresources = `true|false`
    Habilita la función de recursos globales [55]. Valor predeterminado: `false`.
    Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

  - globalresourceconfig [gc]

    --gc | --globalresourceconfig = *VALOR*
    Especifica la configuración activa del recurso global [55] (y habilita los recursos globales) [55].

  - globalresourcefile [gr]

    --gr | --globalresourcefile = *ARCHIVO*
    Especifica el archivo de recursos globales [55] (y habilita los recursos globales) [55].

© 2017-2023 Altova GmbH

5.5.11 xsd2jsonschema

El comando **xsd2jsonschema** convierte uno o varios documentos de esquema 1.0 o 1.1 de XML del W3C en un documento de esquema JSON.

```
raptorxml xsd2jsonschema [opciones] ArchivoXSD
```

- El argumento *ArchivoXSD* es el documento de esquema XML que se quiere convertir.
- Utilice la opción `--schema-conversion-output` para especificar dónde se debe guardar el archivo XSD generado.

Ejemplo

Ejemplo del comando **xsd2jsonschema**:

- **raptorxml** xsd2jsonschema --schema-conversion-output=c:\MiEsquemaJSON.json c:\MiEsquemaXML.xsd

▼ Mayúsculas/minúsculas y barras en la línea de comandos

RaptorXML (y **RaptorXMLServer** para comandos administrativos) *en Windows*
raptorxml (y **raptorxmlserver** para comandos administrativos) *en Windows y Unix (Linux, Mac)*

* Las minúsculas (raptorxml y raptorxmlserver) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que las mayúsculas (RaptorXML) funcionan solamente en Windows y Mac.
* Use la barra diagonal en Linux y Mac y la barra diagonal inversa en Windows.

▼ Barra diagonal inversa y espacios en sistemas Windows

En sistemas Windows: si hay espacios o caracteres especiales (por ejemplo en los nombres de archivos o de carpetas, empresas, personas o productos) debe usar comillas: por ejemplo, **"Mi archivo"**. Sin embargo, debe tener en cuenta que una barra diagonal inversa seguida de comillas dobles (por ejemplo, **"C:\Mi Directorio\"**) es posible que no se lea correctamente. Esto se debe a que la barra diagonal inversa también se usa para empezar secuencias de escape y la secuencia de escape **\"** también usa las comillas dobles. Para evitar secuencia de caracteres puede añadir otra barra diagonal inversa: **\\"**. En resumen: si necesita escribir una ruta que contenga espacios y una barra diagonal inversa, esta es la mejor forma de hacerlo: **"C:\Mi Directorio\\"**.

Opciones

Las opciones del comando aparecen más abajo divididas en grupos. Los valores se pueden dar sin comillas excepto en estos dos casos: (i) cuando la cadena de valor contiene espacios y (ii) cuando en la descripción de la opción se indique explícitamente que es necesario el uso de comillas. Si una opción toma un valor booleano y no se indica ningún valor, entonces el valor predeterminado de la opción es TRUE. Use la opción **--h, --help** para ver información sobre el comando.

▼ XML Schema - opciones de definición

▼ schema-imports

--schema-imports = load-by-schemalocation | load-preferring-schemalocation | load-by-namespace | load-combining-both | license-namespace-only
Esta opción indica el comportamiento de los elementos xs:import. Cada uno de estos elementos tiene un atributo opcional namespace y un atributo opcional schemaLocation: <import namespace="unEspacioNombres" schemaLocation="unaURL">. La opción indica si se debe cargar un documento de esquema o solo autorizar a un espacio de nombres. Si la opción indica que se debe cargar un documento de esquema, entonces indica también qué información debe utilizarse para encontrar el documento de esquema. Valor predeterminado: load-preferring-schemalocation.

- load-by-schemalocation: el valor del atributo schemaLocation se utiliza para buscar el esquema, teniendo en cuenta las asignaciones de catálogo[48]. Si está presente el atributo namespace, se importa el espacio de nombres (con licencia).
- load-preferring-schemalocation: si está presente, se utiliza el atributo schemaLocation teniendo en cuenta las asignaciones de catálogo[48]. Si no está presente el atributo schemaLocation, entonces se usa el valor del atributo namespace a través de las asignaciones de catálogo[48]. Este es el **valor predeterminado**.
- load-by-namespace: el valor del atributo namespace se utiliza para buscar el esquema por medio de una asignación de catálogo[48].
- load-combining-both: si el atributo namespace o schemaLocation tiene una asignación de catálogo[48], entonces se usa la asignación. Si ambos atributos tienen asignaciones de catálogo[48], entonces es el valor de la opción --schema-mapping (opción XML/XSD[240]) decide qué asignación se utiliza. Si no hay ninguna asignación de catálogo[48], entonces se usa el atributo schemaLocation.
- license-namespace-only: se importa el espacio de nombres. No se importa el documento de esquema.

▼ schema-mapping

--schema-mapping = prefer-schemalocation | prefer-namespace
Si se usa la ubicación y el espacio de nombres para buscar el documento de esquema, esta opción indica cuál de ellos debe ser la opción preferida durante la búsqueda en el catálogo.
Si la opción --schemalocation-hints o la opción --schema-imports tiene el valor load-combining-both y si las partes de espacio de nombres y URL pertinentes tienen asignaciones de catálogo[48], entonces el valor de la opción especifica cuál de las dos asignaciones se utiliza (la asignación del espacio de nombres o de la URL: el valor prefer-schemalocation se refiere a la asignación de la URL). Valor predeterminado: prefer-schemalocation.

▼ schemalocation-hints

--schemalocation-hints = load-by-schemalocation | load-by-namespace | load-combining-both | ignore
Determina el comportamiento predeterminado de los atributos xsi:schemaLocation y xsi:noNamespaceSchemaLocation. Indica si se debe cargar un documento de esquema y, si así es, indica qué información debe utilizarse para encontrarlo. Valor predeterminado: load-by-schemalocation.

- **Valor predeterminado**: load-by-schemalocation. Este valor toma la URL de la ubicación del esquema[403] de los atributos xsi:schemaLocation y xsi:noNamespaceSchemaLocation de los documentos de instancia XML.

- El valor load-by-namespace toma la parte de espacio de nombres[403] del atributo

 © 2017-2023 Altova GmbH

`xsi:schemaLocation` y una cadena vacía en el caso del atributo `xsi:noNamespaceSchemaLocation` y encuentra el esquema por medio de una asignación de catálogo[48].

- Si usa el valor `load-combining-both` y el espacio de nombres o la URL tienen una asignación de catálogo[48], se usa dicha asignación. Si ambos tienen asignaciones de catálogo[48], el valor de la opción `schema-mapping` (opción XML/XSD[240]) decide qué asignación se utiliza. Si ni el espacio de nombres ni la URL tiene una asignación de catálogo, se usa la URL.

- El valor `ignore` ignora los atributos `xsi:schemaLocation` y `xsi:noNamespaceSchemaLocation`.

▼ Opciones de procesamiento de esquemas XML

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

`--report-import-namespace-mismatch-as-warning =` `true|false`
Informa de los errores de disparidad del espacio de nombres o del espacio de nombres de destino al importar esquemas con **`xs:import`** como advertencias en vez de como errores. El valor predeterminado es `false`.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

▼ xinclude

`--xinclude =` `true|false`
Habilita la compatibilidad con inclusiones XML (XInclude). Si el valor es `false`, los elementos XInclude `include` se ignoran. Valor predeterminado: `false`.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

▼ xml-mode-for-schemas

`--xml-mode-for-schemas =` `wf|id|valid`
Especifica el modo de procesamiento XML que debe utilizarse para el documento de instancia XML: `wf`=comprobación de formato; `id`=comprobación de formato con ID/IDREF; `valid`=validación. Valor predeterminado: `wf`. Recuerde que el valor `valid` exige que cada documento de esquema que se cargue durante el procesamiento haga referencia a una DTD. Si no existe ninguna DTD, se generará un error.

▼ xsd-version

`--xsd-version =` `1.0|1.1|detect`

Especifica qué versión de la especificación Schema Definition Language (XSD) del W3C se debe usar. Valor predeterminado: `1.0`.

Esta opción también puede ser útil si quiere ver en qué aspectos no es compatible un esquema 1.0 con la especificación 1.1. El valor `detect` es una característica de Altova. Permite detectar la versión del esquema XML (`1.0` o `1.1`) leyendo el valor del atributo `vc:minVersion` del elemento `<xs:schema>` del documento. Si el valor del atributo `@vc:minVersion` es `1.1`, se entiende que la versión del esquema es `1.1`. Si el atributo tiene otro valor que no sea `1.1` (o si no está presente el atributo `@vc:minVersion`), se entiende que la versión del esquema es `1.0`.

- Conversión de archivo XSD a esquema JSON
  - array-and-item

    --array-and-item = `true|false`
    Si está establecido en **true**, el esquema JSON generado permitirá no sólo matrices sino también elementos individuales para partículas con `maxOccurs > 1`. El valor predeterminado para esta opción es **true**.
    Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

  - jsonschema-version

    --jsonschema-version = `draft04|draft06|draft07|2019-09|2020-12|latest|detect`
    Indica qué versión de la versión del borrador de la especificación JSON Schema usar. El valor predeterminado es `detect`.

  - property-for-comments

    --property-for-comments = `true|false`
    Si está configurado en **true**, se crea una propiedad llamada `'#'` en cada subesquema para que se admitan comentarios. El valor predeterminado para esta opción es **false**.
    Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

  - property-for-pis

    --property-for-pis = `true|false`
    Si está establecido en **true**, se crea un patrón de propiedades que combina propiedades prefijadas con `'?'` para admitir instrucciones de procesamiento XML. El valor predeterminado para esta opción es **false**.
    Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

  - property-for-xmlns

    --property-for-xmlns = `true|false`
    Si está establecido en **true**, se crea un patrón de propiedades que combina propiedades prefijadas con `'@xmlns'` en cada subesquema para admitir la declaración de espacios de nombres. El valor predeterminado para esta opción es **false**.
    Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

  - property-for-xsi

    --property-for-xsi = `true|false`
    Si está establecido en **true**, se crea un patrón de propiedades que combina propiedades prefijadas con `'@ xsi'` en cada subesquema para admitir atributos `xsi:*`, como por ejemplo `xsi:schemaLocation`. El valor predeterminado para esta opción es **false**.
    Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

  - simple-content-pure-object

    --simple-content-pure-object= `true|false`

© 2017-2023 Altova GmbH

Si está establecido en **true**, se crea un objeto puro para tipos complejos con contenido simple. El valor predeterminado para esta opción es **false**.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- schema-conversion-output, o

  --o, --schema-converson-output = `FILE`
  Establece el nombre y la ruta de acceso del archivo al que se envía el resultado de la conversión.

- simplify-occurrence-constraints

  --simplify-occurrence-constraints = `true|false`
  Si está establecido en **true**: (i) las especificaciones de aparición en el esquema XML se simplifican a obligatorias u opcionales en el esquema JSON: (ii) los elementos que se repiten en el esquema XML se simplifican a matrices con un número ilimitado de elementos (**maxItems**). El valor predeterminado para esta opción es **false**.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- xmlns

  --xml-mode = `wf|id|valid`
  Especifica asignaciones de prefijos de URI para los espacios de nombres en el esquema XML.

- Opciones comunes
  - error-format

    --error-format = `text|shortxml|longxml`
    Especifica el formato de la salida de error. Los valores posibles son formatos de texto, XML y XML detallado (`longxml`). Valor predeterminado: `text`.

  - error-limit

    --error-limit = *N | unlimited*
    Especifica el límite de errores con un valor comprendido entre `1` y `9999` o `unlimited` (ilimitado). El valor predeterminado es `100`. Cuando se alcanza el límite de error, se detiene la validación. Esta opción es muy práctica a la hora de limitar el uso del procesador durante la validación o transformación.

  - info-limit

    --info-limit = *N | unlimited*
    Indica el límite del mensaje de información dentro del rango `1-65535` or `unlimited`. Si se alcanza el límite de información indicado, el procesamiento continúa pero ya no se informa de más mensajes. El valor predeterminado es `100`.

  - help

    --help
    Muestra el texto de ayuda para el comando. Por ejemplo `valany --h`. (Otra opción es usar el

comando `help` con un argumento. Por ejemplo: `help valany`).

- listfile

  `--listfile =` `true|false`
  Si el valor es `true`, el argumento *`ArchivoEntrada`* del comando se entiende como un archivo de texto que contiene un nombre de archivo por línea. Otra opción es enumerar los archivos en la ILC, separados por un espacio. No obstante, recuerde que las ILC tienen un límite de caracteres. Además, no olvide que la opción `--listfile` solamente afecta a los argumentos y no a las opciones.
  Valor predeterminado: `false`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- log-output

  `--log-output =` *`ARCHIVO`*
  Escribe el registro de salida en la URL de archivo indicada. Compruebe que la ILC tiene permiso de escritura en la ubicación de destino.

- network-timeout

  `--network-timeout =` *`VALOR`*
  Indica el tiempo de espera en milisegundos para operaciones remotas de entrada y salida. Valor predeterminado: `40000`.

- recurse

  `--recurse =` `true|false`
  Esta opción se utiliza para seleccionar ficheros dentro de subdirectorios (incluso en archivos ZIP). Si el valor es `true`, el argumento *`ArchivoEntrada`* del comando seleccionará el fichero seleccionado también en los subdirectorios. Por ejemplo: `"test.zip|zip\test.xml"` seleccionará los ficheros llamados `test.xml` en todos los subdirectorios de la carpeta ZIP. Si quiere puede usar los caracteres comodín `*` y `?`. Por ejemplo: `*.xml` seleccionaría todos los ficheros de la carpeta ZIP que tengan la extensión `.xml`. Valor predeterminado: `false`
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- verbose

  `--verbose =` `true|false`
  Si el valor es `true`, se genera información adicional durante la validación. Valor predeterminado es `false`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- verbose-output

  `--verbose-output =` *`ARCHIVO`*
  Escribe el resultado detallado en el *`ARCHIVO`* indicado.

- version

 © 2017-2023 Altova GmbH

--version
Muestra el número de versión de RaptorXML Server. Si se utiliza con un comando, escriba la opción `--version` antes del comando.

- warning-limit

--warning-limit = *N | unlimited*
Especifica el límite de advertencia en el rango `1-65535` o `unlimited` (ilimitado). El procesamiento continua si se alcanza el límite pero no se registrarán más advertencias. El valor por defecto es `100`.

- Catálogos y recursos globales
  - catalog

--catalog = *ARCHIVO*
Especifica la ruta de acceso absoluta a un archivo de catálogo que no está en el archivo de catálogo raíz instalado. El valor predeterminado es la ruta de acceso absoluta del archivo de catálogo raíz instalado (`<carpeta-instalación>\Altova\RaptorXMLServer2024\etc\RootCatalog.xml`). Consulte el apartado Catálogos XML [48] para obtener más información.

  - user-catalog

--user-catalog = *ARCHIVO*
Especifica la ruta de acceso absoluta a un catálogo XML que debe utilizarse junto con el catálogo raíz. Consulte el apartado Catálogos XML [48] para obtener más información.

  - enable-globalresources

--enable-globalresources = `true|false`
Habilita la función de recursos globales [55]. Valor predeterminado: `false`.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

  - globalresourceconfig [gc]

--gc | --globalresourceconfig = *VALOR*
Especifica la configuración activa del recurso global [55] (y habilita los recursos globales) [55].

  - globalresourcefile [gr]

--gr | --globalresourcefile = *ARCHIVO*
Especifica el archivo de recursos globales [55] (y habilita los recursos globales) [55].

5.6 Comandos XML Signature

Los comandos XML Signature sirven para firmar documentos XML y verificar documentos firmados. Encontrará una descripción detallada de los comandos XML Signature en los apartados de esta sección:

xmlsignature-sign 196	Crea un documento de salida de firma XML a partir de un documento de entrada.
xmlsignature-verify 201	Verifica un documento de firma XML.
xmlsignature-update 204	Actualiza la firma de un documento XML (modificado).
xmlsignature-remove 207	Quita la firma de un documento XML.

5.6.1 xmlsignature-sign

El comando **xmlsignature-sign | xsign** toma como entrada un documento XML y crea un documento con firma XML de salida usando las opciones de firma especificadas.

Windows
```
RaptorXML xmlsignature-sign [opciones] --output=Archivo --signature-type=Valor
--signature-canonicalization-method=Valor --certname=Value|hmackey=Valor
ArchivoEntrada
```

Linux
```
raptorxml xmlsignature-sign [opciones] --output=Archivo --signature-type=Valor
--signature-canonicalization-method=Valor --certname=Value|hmackey=Valor
ArchivoEntrada
```

Mac
```
raptorxml xmlsignature-sign [opciones] --output=Archivo --signature-type=Valor
--signature-canonicalization-method=Valor --certname=Value|hmackey=Valor
ArchivoEntrada
```

El argumento *ArchivoEntrada* es el documento XML que se debe firmar. La opción **--output** especifica la ubicación del documento que contiene la firma XML.

Ejemplos

- **raptorxml** xsign --output=c:\ArchivoFirmado.xml --signature-type=enveloped --signature-canonicalization-method=xml-c14n11 c:\ArchivoSinFirmar.xml

▼ Mayúsculas/minúsculas y barras en la línea de comandos

RaptorXML (y **RaptorXMLServer** para comandos administrativos) *en Windows*
raptorxml (y **raptorxmlserver** para comandos administrativos) *en Windows y Unix (Linux, Mac)*

 © 2017-2023 Altova GmbH

* Las minúsculas (`raptorxml` y `raptorxmlserver`) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que las mayúsculas (`RaptorXML`) funcionan solamente en Windows y Mac.
* Use la barra diagonal en Linux y Mac y la barra diagonal inversa en Windows.

▼ Barra diagonal inversa y espacios en sistemas Windows

En sistemas Windows: si hay espacios o caracteres especiales (por ejemplo en los nombres de archivos o de carpetas, empresas, personas o productos) debe usar comillas: por ejemplo, **`"Mi archivo"`**. Sin embargo, debe tener en cuenta que una barra diagonal inversa seguida de comillas dobles (por ejemplo, **`"C:\Mi Directorio\"`**) es posible que no se lea correctamente. Esto se debe a que la barra diagonal inversa también se usa para empezar secuencias de escape y la secuencia de escape **`\"`** también usa las comillas dobles. Para evitar secuencia de caracteres puede añadir otra barra diagonal inversa: **`\\"`**. En resumen: si necesita escribir una ruta que contenga espacios y una barra diagonal inversa, esta es la mejor forma de hacerlo: **`"C:\Mi Directorio\\"`**.

Opciones

Las opciones del comando aparecen más abajo divididas en grupos. Los valores se pueden dar sin comillas excepto en estos dos casos: (i) cuando la cadena de valor contiene espacios y (ii) cuando en la descripción de la opción se indique explícitamente que es necesario el uso de comillas. Si una opción toma un valor booleano y no se indica ningún valor, entonces el valor predeterminado de la opción es `TRUE`. Use la opción **`--h, --help`** para ver información sobre el comando.

▼ Opciones comunes

▼ output

`output =` *ARCHIVO*
La URL del documento de salida que se crea con la nueva firma XML.

▼ verbose

`--verbose =` `true|false`
Si el valor es `true`, se genera información adicional durante la validación. Valor predeterminado es `false`.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

▼ Opciones de firma XML

▼ absolute-reference-uri

`--absolute-reference-uri =` `true|false`
Especifica si el URI del documento firmado debe leerse como absoluto (`true`) o relativo (`false`). El valor predeterminado es `false`.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

▼ certname, certificate-name

`--certname, --certificate-name =` *VALOR*

El nombre del certificado utilizado para firmar.

Windows
Se trata del nombre `Subject` de un certificado del almacén de certificados (**`--certificate-store`**) seleccionado.

iemplo para enumerar los certificados (bajo PowerShell)

```
% ls cert://UsuarioActual/My
PSParentPath: Microsoft.PowerShell.Security\Certificate::UsuarioActual\My
Thumbprint Subject
---------- -------
C9DF64BB0AAF5FA73474D78B7CCFFC37C95BFC6C CN=certificate1
... CN=...
```

iemplo: **`--certificate-name==certificado1`**

Linux/macOS
`--certname` especifica el nombre de archivo de un certificado X.509v3 con cifrado PEM con la clave privada. Estos archivos suelen tener la extensión **`.pem`**.

iemplo: **`--certificate-name==/ruta/de/certificado1.pem`**

▼ certstore, certificate-store

`--certstore, --certificate-store =` *VALOR*
La ubicación donde está almacenado el certificado especificado con **`--certificate-name`**.

Windows
Nombre de un almacén de certificados situado dentro de **`cert://UsuarioActual`**. Los almacenes de certificados disponibles se pueden enumerar (bajo PowerShell) usando **`% ls cert://UsuarioActual/`**. Los certificados se enumerarán de la siguiente manera:

```
Nombre : TrustedPublisher
Nombre : ClientAuthIssuer
Nombre : Root
Nombre : UserDS
Nombre : CA
Nombre : ACRS
Nombre : REQUEST
Nombre : AuthRoot
Nombre : MSIEHistoryJournal
Nombre : TrustedPeople
Nombre : MyCertStore
Nombre : Local NonRemovable Certificates
Nombre : SmartCardRoot
Nombre : Trust
Nombre : Disallowed
```

 © 2017-2023 Altova GmbH

iemplo: **--certificate-store==MiAlmacénCertificados**

Linux/MacOS
La opción **--certstore** no es compatible por ahora.

▼ digest, digest-method

--digest, --digest-method = sha1|sha256|sha384|sha512
El algoritmo que se usa para computar el valor del resumen (digest) dentro del archivo XML de entrada. Los valores disponibles son: sha1|sha256|sha384|sha512.

▼ hmackey, hmac-secret-key

--hmackey, --hmac-secret-key = *VALOR*
La clave secreta compartida HMAC. Debe tener una longitud mínima de seis caracteres.

Ejemplo: **--hmackey=contraseñaSecreta**

▼ hmaclen, hmac-output-length

--hmaclen, --hmac-output-length = *LONGITUD*

Trunca el resultado del algoritmo HMAC hasta el número de bits indicado por LONGITUD. Si se especifica, este valor debe ser:
- múltiplo de 8,
- mayor que 80 y
- mayor que la mitad de la longitud de salida del algoritmo hash subyacente.

▼ keyinfo, append-keyinfo

--keyinfo, --append-keyinfo = true|false
Especifica si se debe incluir el elemento **KeyInfo** en la firma o no. El valor predeterminado es **false**.

▼ sigc14nmeth, signature-canonicalization-method

--sigc14nmeth, --signature-canonicalization-method = *VALOR*
Especifica el algoritmo de canonización que se debe aplicar al elemento **SignedInfo**. El valor debe ser uno de estos tres valores:

- REC-xml-c14n-20010315
- xml-c14n11
- xml-exc-c14n#

▼ sigmeth, signature-method

--sigmeth, --signature-method = *VALOR*
Especifica el algoritmo que se debe usar para generar la firma.

Cuando se usa un certificado
Si se especificó un certificado, entonces `--signature-method` es opcional y el valor para este parámetro se deriva del certificado. Debe coincidir con el algoritmo utilizado por el certificado.

iemplo: **`--signature-method=rsa-sha256`**

Cuando se usa --hmac-secret-key
Si se usa la opción `--hmac-secret-key`, entonces esta opción es obligatoria. El valor debe ser uno de los algoritmos HMAC compatibles:

- `hmac-sha256`
- `hmac-sha386`
- `hmac-sha512`
- `hmac-sha1` (no recomendable según la especificación)

iemplo: **`--signature-method=hmac-sha256`**

▼ sigtype, signature-type

`--sigtype, --signature-type =` `detached | enveloping | enveloped`
Especifica el tipo de firma que se debe generar (separada, envolvente o envuelta).

▼ transforms

`--transforms =` *VALOR*
Especifica las transformaciones XML Signature que se deben aplicar al documento de salida. Esta opción se puede especificar más de una vez. Si se especifica varias veces, el orden en que se especifica es importante. La primera transformación especificada recibe el documento de entrada. La última se utiliza inmediatamente antes de calcular el valor implícito.

Valores compatibles:

- `REC-xml-c14n-20010315` para Canonical XML 1.0 (omitir comentarios)
- `xml-c14n11` para Canonical XML 1.1 (omitir comentarios)
- `xml-exc-c14n#` para Exclusive XML Canonicalization 1.0 (omitir comentarios)
- `REC-xml-c14n-20010315#WithComments` para Canonical XML 1.0 (con comentarios)
- `xml-c14n11#WithComments` para Canonical XML 1.1 (con comentarios)
- `xml-exc-c14n#WithComments` para Exclusive XML Canonicalization 1.0 (con comentarios)
- `base64`
- extensión de Altova `strip-whitespaces`

iemplo: **`--transforms=xml-c14n11`**

▼ write-default-attributes

`--write-default-attributes =` `true|false`
Especifica si se deben incluir los valores de atributo predeterminados de la DTD en el documento firmado.

▼ Opciones de ayuda y versión

▼ help

 © 2017-2023 Altova GmbH

--help
Muestra el texto de ayuda para el comando. Por ejemplo `valany --h`. (Otra opción es usar el comando `help` con un argumento. Por ejemplo: `help valany`).

- version

  --version
  Muestra el número de versión de RaptorXML Server. Si se utiliza con un comando, escriba la opción `--version` antes del comando.

5.6.2 xmlsignature-verify

El comando **xmlsignature-verify | xverify** verifica la firma XML del archivo de entrada. Si la firma se verifica correctamente, aparece un mensaje **resultado="OK"**. En caso contrario aparece el mensaje **resultado="Con errores"**.

Windows	**RaptorXML xmlsignature-verify [opciones] *ArchivoEntrada***
Linux	**raptorxml xmlsignature-verify [opciones] *ArchivoEntrada***
Mac	**raptorxml xmlsignature-verify [opciones] *ArchivoEntrada***

El argumento *ArchivoEntrada* es el documento XML firmado que se debe verificar.

Ejemplos

- **raptorxml** xverify c:\ArchivoFirmado.xml

- Mayúsculas/minúsculas y barras en la línea de comandos

  RaptorXML (y **RaptorXMLServer** para comandos administrativos) *en Windows*
  raptorxml (y **raptorxmlserver** para comandos administrativos) *en Windows y Unix (Linux, Mac)*

  * Las minúsculas (`raptorxml` y `raptorxmlserver`) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que las mayúsculas (`RaptorXML`) funcionan solamente en Windows y Mac.
  * Use la barra diagonal en Linux y Mac y la barra diagonal inversa en Windows.

- Barra diagonal inversa y espacios en sistemas Windows

  En sistemas Windows: si hay espacios o caracteres especiales (por ejemplo en los nombres de archivos o de carpetas, empresas, personas o productos) debe usar comillas: por ejemplo, **"Mi archivo"**. Sin embargo, debe tener en cuenta que una barra diagonal inversa seguida de comillas dobles (por ejemplo, **"C:\Mi Directorio\"**) es posible que no se lea correctamente. Esto se debe a que la barra diagonal inversa también se usa para empezar secuencias de escape y la secuencia de escape **\"** también usa las comillas dobles. Para evitar secuencia de caracteres puede añadir otra barra diagonal inversa: **\\"**. En

resumen: si necesita escribir una ruta que contenga espacios y una barra diagonal inversa, esta es la mejor forma de hacerlo: `"C:\Mi Directorio\\"`.

Opciones

Las opciones del comando aparecen más abajo divididas en grupos. Los valores se pueden dar sin comillas excepto en estos dos casos: (i) cuando la cadena de valor contiene espacios y (ii) cuando en la descripción de la opción se indique explícitamente que es necesario el uso de comillas. Si una opción toma un valor booleano y no se indica ningún valor, entonces el valor predeterminado de la opción es `TRUE`. Use la opción **`--h, --help`** para ver información sobre el comando.

- Opciones comunes
  - verbose

    `--verbose =` `true|false`
    Si el valor es `true`, se genera información adicional durante la validación. Valor predeterminado es `false`.
    Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- Opciones de firma XML
  - certname, certificate-name

    `--certname, --certificate-name =` *VALOR*
    El nombre del certificado utilizado para firmar.

    Windows
    Se trata del nombre `Subject` de un certificado del almacén de certificados (**`--certificate-store`**) seleccionado.

    Ejemplo para enumerar los certificados (bajo PowerShell)

```
% ls cert://UsuarioActual/My
PSParentPath: Microsoft.PowerShell.Security\Certificate::UsuarioActual\My
Thumbprint Subject
---------- -------
C9DF64BB0AAF5FA73474D78B7CCFFC37C95BFC6C CN=certificate1
... CN=...
```

    Ejemplo: **`--certificate-name==certificado1`**

    Linux/macOS
    `--certname` especifica el nombre de archivo de un certificado X.509v3 con cifrado PEM con la clave privada. Estos archivos suelen tener la extensión **`.pem`**.

    Ejemplo: **`--certificate-name==/ruta/de/certificado1.pem`**

  - certstore, certificate-store

 © 2017-2023 Altova GmbH

--certstore, --certificate-store = *VALOR*
La ubicación donde está almacenado el certificado especificado con **--certificate-name**.

Windows
Nombre de un almacén de certificados situado dentro de **cert://UsuarioActual**. Los almacenes de certificados disponibles se pueden enumerar (bajo PowerShell) usando **% ls cert://UsuarioActual/**. Los certificados se enumerarán de la siguiente manera:

```
Nombre : TrustedPublisher
Nombre : ClientAuthIssuer
Nombre : Root
Nombre : UserDS
Nombre : CA
Nombre : ACRS
Nombre : REQUEST
Nombre : AuthRoot
Nombre : MSIEHistoryJournal
Nombre : TrustedPeople
Nombre : MyCertStore
Nombre : Local NonRemovable Certificates
Nombre : SmartCardRoot
Nombre : Trust
Nombre : Disallowed
```

iemplo: **--certificate-store==MiAlmacénCertificados**

Linux/MacOS
La opción **--certstore** no es compatible por ahora.

- hmackey, hmac-secret-key

  --hmackey, --hmac-secret-key = *VALOR*
  La clave secreta compartida HMAC. Debe tener una longitud mínima de seis caracteres.

  Ejemplo: **--hmackey=contraseñaSecreta**

- Opciones de ayuda y versión
  - help

    --help
    Muestra el texto de ayuda para el comando. Por ejemplo `valany --h`. (Otra opción es usar el comando `help` con un argumento. Por ejemplo: `help valany`).

  - version

    --version
    Muestra el número de versión de RaptorXML Server. Si se utiliza con un comando, escriba la opción `--version` antes del comando.

5.6.3 xmlsignature-update

El comando **xmlsignature-update | xupdate** actualiza la firma XML en el archivo de entrada firmado. Si el documento sufrió cambios, la firma XML será diferente. En caso contrario, la firma actualizada será la misma que la anterior.

Windows	**RaptorXML xmlsignature-update [opciones] --output=Archivo *ArchivoFirmado***
Linux	**raptorxml xmlsignature-update [opciones] --output=Archivo *ArchivoFirmado***
Mac	**raptorxml xmlsignature-update [opciones] --output=Archivo *ArchivoFirmado***

El argumento *ArchivoFirmado* es el documento XML firmado que se debe actualizar. Es obligatorio especificar (i) la opción `hmac-secret-key` o (ii) la opción `certificate-name`. Si se especifican las opciones `certificate-name` y `certificate-store` , entonces deben coincidir con las utilizadas previamente para firmar el documento XML. (Recuerde que la opción `certificate-store` no se puede usar en Linux ni macOS.)

Ejemplos

- **raptorxml** xupdate --output=c:\ActualizarArchivoFirmado.xml --certname=certificado1 --certstore=MiAlmacénCert c:\UnArchivoFirmado.xml
- **raptorxml** xupdate --output=c:\ActualizarArchivoFirmado.xml --hmackey=ContraseñaSecreta c:\UnArchivoFirmado.xml

▼ Mayúsculas/minúsculas y barras en la línea de comandos

RaptorXML (y **RaptorXMLServer** para comandos administrativos) *en Windows*
raptorxml (y **raptorxmlserver** para comandos administrativos) *en Windows y Unix (Linux, Mac)*

* Las minúsculas (`raptorxml` y `raptorxmlserver`) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que las mayúsculas (`RaptorXML`) funcionan solamente en Windows y Mac.
* Use la barra diagonal en Linux y Mac y la barra diagonal inversa en Windows.

▼ Barra diagonal inversa y espacios en sistemas Windows

En sistemas Windows: si hay espacios o caracteres especiales (por ejemplo en los nombres de archivos o de carpetas, empresas, personas o productos) debe usar comillas: por ejemplo, **"Mi archivo"**. Sin embargo, debe tener en cuenta que una barra diagonal inversa seguida de comillas dobles (por ejemplo, **"C:\Mi Directorio\"**) es posible que no se lea correctamente. Esto se debe a que la barra diagonal inversa también se usa para empezar secuencias de escape y la secuencia de escape **\"** también usa las comillas dobles. Para evitar secuencia de caracteres puede añadir otra barra diagonal inversa: **\\"**. En resumen: si necesita escribir una ruta que contenga espacios y una barra diagonal inversa, esta es la mejor forma de hacerlo: **"C:\Mi Directorio\\"**.

 © 2017-2023 Altova GmbH

Opciones

Las opciones del comando aparecen más abajo divididas en grupos. Los valores se pueden dar sin comillas excepto en estos dos casos: (i) cuando la cadena de valor contiene espacios y (ii) cuando en la descripción de la opción se indique explícitamente que es necesario el uso de comillas. Si una opción toma un valor booleano y no se indica ningún valor, entonces el valor predeterminado de la opción es `TRUE`. Use la opción **`--h, --help`** para ver información sobre el comando.

- Opciones comunes
  - output

    `output =` *ARCHIVO*
    La URL del documento de salida que se crea con la nueva firma XML.

  - verbose

    `--verbose =` `true|false`
    Si el valor es `true`, se genera información adicional durante la validación. Valor predeterminado es `false`.
    Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- Opciones de firma XML
  - certname, certificate-name

    `--certname, --certificate-name =` *VALOR*
    El nombre del certificado utilizado para firmar.

    Windows
    Se trata del nombre `Subject` de un certificado del almacén de certificados (**`--certificate-store`**) seleccionado.

    Ejemplo para enumerar los certificados (bajo PowerShell)

```
% ls cert://UsuarioActual/My
PSParentPath: Microsoft.PowerShell.Security\Certificate::UsuarioActual\My
Thumbprint Subject
---------- -------
C9DF64BB0AAF5FA73474D78B7CCFFC37C95BFC6C CN=certificate1
... CN=...
```

    Ejemplo: **`--certificate-name==certificado1`**

    Linux/macOS
    `--certname` especifica el nombre de archivo de un certificado X.509v3 con cifrado PEM con la clave privada. Estos archivos suelen tener la extensión **`.pem`**.

    Ejemplo: **`--certificate-name==/ruta/de/certificado1.pem`**

- certstore, certificate-store

  --certstore, --certificate-store = *VALOR*
  La ubicación donde está almacenado el certificado especificado con **--certificate-name**.

  Windows
  Nombre de un almacén de certificados situado dentro de **cert://UsuarioActual**. Los almacenes de certificados disponibles se pueden enumerar (bajo PowerShell) usando **% ls cert://UsuarioActual/**. Los certificados se enumerarán de la siguiente manera:

```
Nombre : TrustedPublisher
Nombre : ClientAuthIssuer
Nombre : Root
Nombre : UserDS
Nombre : CA
Nombre : ACRS
Nombre : REQUEST
Nombre : AuthRoot
Nombre : MSIEHistoryJournal
Nombre : TrustedPeople
Nombre : MyCertStore
Nombre : Local NonRemovable Certificates
Nombre : SmartCardRoot
Nombre : Trust
Nombre : Disallowed
```

  iemplo: **--certificate-store==MiAlmacénCertificados**

  Linux/MacOS
  La opción **--certstore** no es compatible por ahora.

- hmackey, hmac-secret-key

  --hmackey, --hmac-secret-key = *VALOR*
  La clave secreta compartida HMAC. Debe tener una longitud mínima de seis caracteres.

  Ejemplo: **--hmackey=contraseñaSecreta**

- Opciones de ayuda y versión
  - help

    --help
    Muestra el texto de ayuda para el comando. Por ejemplo `valany --h`. (Otra opción es usar el comando `help` con un argumento. Por ejemplo: `help valany`).

  - version

    --version
    Muestra el número de versión de RaptorXML Server. Si se utiliza con un comando, escriba la opción `--version` antes del comando.

 © 2017-2023 Altova GmbH

5.6.4 xmlsignature-remove

El comando **xmlsignature-remove | xremove** quita la firma XML del archivo de entrada firmado y guarda el documento de salida sin firmar en la ubicación de destino especificada.

Windows	**RaptorXML xmlsignature-remove [opciones] --output=Archivo *ArchivoFirmado***
Linux	**raptorxml xmlsignature-remove [opciones] --output=Archivo *ArchivoFirmado***
Mac	**raptorxml xmlsignature-remove [opciones] --output=Archivo *ArchivoFirmado***

El argumento *ArchivoFirmado* es el documento XML firmado del que desea quitar la firma XML. La opción `--output` especifica la ubicación del documento XML sin firmar que se genera.

Ejemplos

- **raptorxml** xremove --output=c:\ArchivoSinFirmar.xml c:\ArchivoFirmado.xml

▼ Mayúsculas/minúsculas y barras en la línea de comandos

RaptorXML (y **RaptorXMLServer** para comandos administrativos) *en Windows*
raptorxml (y **raptorxmlserver** para comandos administrativos) *en Windows y Unix (Linux, Mac)*

* Las minúsculas (`raptorxml` y `raptorxmlserver`) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que las mayúsculas (`RaptorXML`) funcionan solamente en Windows y Mac.
* Use la barra diagonal en Linux y Mac y la barra diagonal inversa en Windows.

▼ Barra diagonal inversa y espacios en sistemas Windows

En sistemas Windows: si hay espacios o caracteres especiales (por ejemplo en los nombres de archivos o de carpetas, empresas, personas o productos) debe usar comillas: por ejemplo, **"Mi archivo"**. Sin embargo, debe tener en cuenta que una barra diagonal inversa seguida de comillas dobles (por ejemplo, **"C:\Mi Directorio\"**) es posible que no se lea correctamente. Esto se debe a que la barra diagonal inversa también se usa para empezar secuencias de escape y la secuencia de escape **\"** también usa las comillas dobles. Para evitar secuencia de caracteres puede añadir otra barra diagonal inversa: **\\"**. En resumen: si necesita escribir una ruta que contenga espacios y una barra diagonal inversa, esta es la mejor forma de hacerlo: **"C:\Mi Directorio\\"**.

Opciones

Las opciones del comando aparecen más abajo divididas en grupos. Los valores se pueden dar sin comillas excepto en estos dos casos: (i) cuando la cadena de valor contiene espacios y (ii) cuando en la descripción de la opción se indique explícitamente que es necesario el uso de comillas. Si una opción toma un valor booleano y no se indica ningún valor, entonces el valor predeterminado de la opción es `TRUE`. Use la opción **--h, --**

help para ver información sobre el comando.

- Opciones comunes
  - output

    output = *ARCHIVO*
    La URL del archivo de salida que se crea tras eliminar la firma XML.

  - verbose

    --verbose = `true|false`
    Si el valor es `true`, se genera información adicional durante la validación. Valor predeterminado es `false`.
    Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- Opciones de ayuda y versión
  - help

    --help
    Muestra el texto de ayuda para el comando. Por ejemplo `valany --h`. (Otra opción es usar el comando `help` con un argumento. Por ejemplo: `help valany`).

  - version

    --version
    Muestra el número de versión de RaptorXML Server. Si se utiliza con un comando, escriba la opción `--version` antes del comando.

© 2017-2023 Altova GmbH

5.7 Comandos generales

En este apartado puede consultar las descripciones de estos comandos generales:

valany 209	Valida el documento enviado según su tipo.
scrip 210	Ejecuta un script Python.
help 211	Muestra información sobre un comando en concreto.

5.7.1 valany

El comando **valany** es un comando de uso general que valida un documento en función de su tipo. El tipo de documento de entrada se detecta automáticamente y la correspondiente validación se lleva a cabo según lo dispuesto por la especificación pertinente. El argumento *ArchivoEntrada* es el documento que se debe validar. Recuerde que como puede dar un documento como máximo como argumento del comando.

Windows **RaptorXML valany [opciones] *ArchivoEntrada***

Linux **raptorxml valany [opciones] *ArchivoEntrada***

Mac **raptorxml valany [opciones] *ArchivoEntrada***

El comando **valany** se encarga de los tipos de validación que se enumeran a continuación. Sus opciones son todas las opciones disponibles para cada uno de los comandos de validación correspondientes. Consulte la descripción de cada comando de validación para ver su lista de opciones.

- valdtd (dtd) 71
- valxsd (xsd) 75
- valxml-withdtd (xml) 59
- valxml-withxsd (xsi) 63
- valxslt 137
- valxquery 114
- valavrojson (avrojson) 162

Ejemplos

- **raptorxmlxbrl** valany c:\Test.xsd

▼ Mayúsculas/minúsculas y barras en la línea de comandos

RaptorXML (y **RaptorXMLServer** para comandos administrativos) *en Windows*
raptorxml (y **raptorxmlserver** para comandos administrativos) *en Windows y Unix (Linux, Mac)*

* Las minúsculas (`raptorxml` y `raptorxmlserver`) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que las mayúsculas (`RaptorXML`) funcionan solamente en Windows y Mac.
* Use la barra diagonal en Linux y Mac y la barra diagonal inversa en Windows.

▼ Barra diagonal inversa y espacios en sistemas Windows

En sistemas Windows: si hay espacios o caracteres especiales (por ejemplo en los nombres de archivos o de carpetas, empresas, personas o productos) debe usar comillas: por ejemplo, **"Mi archivo"**. Sin embargo, debe tener en cuenta que una barra diagonal inversa seguida de comillas dobles (por ejemplo, **"C:\Mi Directorio\"**) es posible que no se lea correctamente. Esto se debe a que la barra diagonal inversa también se usa para empezar secuencias de escape y la secuencia de escape **\"** también usa las comillas dobles. Para evitar secuencia de caracteres puede añadir otra barra diagonal inversa: **\\"**. En resumen: si necesita escribir una ruta que contenga espacios y una barra diagonal inversa, esta es la mejor forma de hacerlo: **"C:\Mi Directorio\\"**.

Opciones

Consulte la descripción de cada comando de validación para ver su lista de opciones. Es importante recordar que, aunque la mayoría de los comandos de validación aceptan varios documentos de entrada, el comando **valany** acepta un documento de entrada como máximo. Por tanto, opciones como `--listfile` no son pertinentes en el caso del comando **valany**.

5.7.2 script

El comando **script** ejecuta un script de Python 3.11.5 que utiliza la API de Python de RaptorXML.

```
raptorxml script [opciones] Archivo de script de Python
```

El argumento *Archivo* es la ruta de acceso del script de Python que se debe ejecutar. Hay varias opciones más, que puede consultar con este comando:

```
raptorxml script [-h | --help]
```

Ejemplos

- **raptorxml** script c:\MiScriptPython.py
- **raptorxml** script -h
- **raptorxml** script # *Sin un archivo de script se inicia una shell de Python interactiva*
- **raptorxml** script -m pip # *Carga y ejecuta el módulo* pip. *Ver apartado "Opciones" más abajo*

▼ Mayúsculas/minúsculas y barras en la línea de comandos

RaptorXML (y **RaptorXMLServer** para comandos administrativos) *en Windows*
raptorxml (y **raptorxmlserver** para comandos administrativos) *en Windows y Unix (Linux, Mac)*

* Las minúsculas (raptorxml y raptorxmlserver) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que las mayúsculas (RaptorXML) funcionan solamente en Windows y Mac.
* Use la barra diagonal en Linux y Mac y la barra diagonal inversa en Windows.

 © 2017-2023 Altova GmbH

- Barra diagonal inversa y espacios en sistemas Windows

  En sistemas Windows: si hay espacios o caracteres especiales (por ejemplo en los nombres de archivos o de carpetas, empresas, personas o productos) debe usar comillas: por ejemplo, `"Mi archivo"`. Sin embargo, debe tener en cuenta que una barra diagonal inversa seguida de comillas dobles (por ejemplo, `"C:\Mi Directorio\"`) es posible que no se lea correctamente. Esto se debe a que la barra diagonal inversa también se usa para empezar secuencias de escape y la secuencia de escape `\"` también usa las comillas dobles. Para evitar secuencia de caracteres puede añadir otra barra diagonal inversa: `\\"`. En resumen: si necesita escribir una ruta que contenga espacios y una barra diagonal inversa, esta es la mejor forma de hacerlo: `"C:\Mi Directorio\\"`.

Opciones

Las opciones y argumentos dados tras el comando `script` se reenvian al intérprete de Python directamente. Consulte la página de la documentación de Python para ver una lista de todas las opciones disponibles (https://docs.python.org/3.11/using/cmdline.html).

5.7.3 help

Sintaxis y descripción

El comando **`help`** toma un único argumento (`Command`), que es el nombre del comando para el que necesita la ayuda, y muestra la sintaxis del comando, sus opciones y otra información relevante. Si no se especifica el comando `Command`, entonces se enumeran todos los comandos del ejecutable, cada uno con una breve descripción.Puede llamar al comando **`help`** desde estos ejecutables: **`raptorxml`** o **`raptorxmlserver`**.

```
raptorxml help Command
raptorxmlserver help Command
```

- Mayúsculas/minúsculas y barras en la línea de comandos

  `RaptorXML` (y **`RaptorXMLServer`** para comandos administrativos) *en Windows*
  `raptorxml` (y **`raptorxmlserver`** para comandos administrativos) *en Windows y Unix (Linux, Mac)*

  * Las minúsculas (`raptorxml` y `raptorxmlserver`) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que las mayúsculas (`RaptorXML`) funcionan solamente en Windows y Mac.
  * Use la barra diagonal en Linux y Mac y la barra diagonal inversa en Windows.

Ejemplo

Ejemplos del comando **`help`** para mostrar información sobre el comando **`licenseserver`** (este comando está en los dos ejecutables):

```
raptorxml help licenseserver
raptorxmlserver help licenseserver
```

La opción --help

También puede acceder a la información de ayuda sobre un comando usando la opción `--help` del comando para el que necesita la ayuda. Los comandos siguientes obtienen el mismo resultado:

```
raptorxml licenseserver --help
```

El comando anterior usa la opción `--help` del comando **`licenseserver`**.

```
raptorxml help licenseserver
```

El comando **`help`** toma como argumento `licenseserver`.

En ambos casos, aparece información de ayuda sobre el comando **`licenseserver`**.

© 2017-2023 Altova GmbH

5.8 Comandos de localización

Si quiere puede localizar (traducir) la aplicación RaptorXML en cualquier idioma. RaptorXML ya está disponible en cinco idiomas (español, francés, inglés, alemán y japonés) en la carpeta `<CarpetaArchivosPrograma>\Altova\RaptorXMLServer2024\bin\`.

Para localizar RaptorXML en otros idiomas:

1. Genere un archivo XML con las cadenas de recursos usando el comando `exportresourcestrings` 213 . Las cadenas de recursos del archivo XML generado estarán en uno de los cinco idiomas compatibles: inglés (`en`), español (`es`), francés (`fr`), alemán (`de`) o japonés (`ja`), dependiendo del argumento utilizado con el comando.
2. Traduzca las cadenas de recursos del XML generado al idioma de destino. Las cadenas de recursos son el contenido de los elementos `<string>` del archivo XML. No traduzca las variables que aparecen entre llaves, como `{option}` o `{product}`, por ejemplo.
3. Póngase en contacto con el equipo de soporte técnico de Altova para generar un archivo DLL localizado de RaptorXML a partir de su archivo XML traducido.
4. Cuando reciba el archivo DLL localizado del equipo de soporte técnico de Altova, guárdelo en la carpeta `<CarpetaArchivosPrograma>\Altova\RaptorXMLServer2024\bin\`. El DLL tendrá un nombre similar a este `RaptorXMLServer_ci.dll`. La parte `_ci` del nombre contiene el código del idioma. Por ejemplo, en `RaptorXMLServer_de.dll`, la parte `de` es el código del idioma alemán (`Deutsch`).
5. Ejecute el comando `setdeflang` 215 para establecer el archivo DLL localizado como aplicación predeterminada. Use el código de idioma del nombre del archivo DLL como argumento del comando `setdeflang` 215.

Nota: Altova RaptorXML Server está disponible en cinco idiomas (español, francés, inglés, alemán y japonés) y, por tanto, no es necesario traducirlo a estos idiomas. Para establecer uno de estos idiomas como idioma predeterminado use el comando `setdeflang` 215.

5.8.1 exportresourcestrings

Sintaxis y descripción

El comando `exportresourcestrings` genera un archivo XML que contiene todas las cadenas de recursos de la aplicación RaptorXML Server en el idioma indicado. Los idiomas en los que se puede generar el archivo son inglés (`en`), español (`es`), francés (`fr`) alemán (`de`) y japonés (`ja`).

```
raptorxml exportresourcestrings [opciones] LanguageCode XMLOutputFile
raptorxmlserver exportresourcestrings [opciones] LanguageCode XMLOutputFile
```

- El argumento ***LanguageCode*** indica el idioma de las cadenas de recursos del archivo XML de salida; se trata del *lenguaje de exportación*. Se admiten estos idiomas de exportación (se indica el código correspondiente entre paréntesis): inglés (`en`), alemán, (`de`), español (`es`), francés (`fr`) y japonés (`ja`).
- El argumento ***XMLOutputFile*** indica la ruta y el nombre del archivo XML de salida.
- Puede llamar al comando **`exportresourcestrings`** desde desde estos dos ejecutables: **`raptorxml`** o **`raptorxmlserver`**.

A continuación explicamos cómo localizar.

▼ Mayúsculas/minúsculas y barras en la línea de comandos

RaptorXML (y **RaptorXMLServer** para comandos administrativos) *en Windows*
raptorxml (y **raptorxmlserver** para comandos administrativos) *en Windows y Unix (Linux, Mac)*

* Las minúsculas (raptorxml y raptorxmlserver) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que las mayúsculas (RaptorXML) funcionan solamente en Windows y Mac.
* Use la barra diagonal en Linux y Mac y la barra diagonal inversa en Windows.

▼ Barra diagonal inversa y espacios en sistemas Windows

En sistemas Windows: si hay espacios o caracteres especiales (por ejemplo en los nombres de archivos o de carpetas, empresas, personas o productos) debe usar comillas: por ejemplo, **"Mi archivo"**. Sin embargo, debe tener en cuenta que una barra diagonal inversa seguida de comillas dobles (por ejemplo, **"C:\Mi Directorio\"**) es posible que no se lea correctamente. Esto se debe a que la barra diagonal inversa también se usa para empezar secuencias de escape y la secuencia de escape **\"** también usa las comillas dobles. Para evitar secuencia de caracteres puede añadir otra barra diagonal inversa: **\\"**. En resumen: si necesita escribir una ruta que contenga espacios y una barra diagonal inversa, esta es la mejor forma de hacerlo: **"C:\Mi Directorio\\"**.

Ejemplos

Ejemplos del comando exportresourcestrings:

```
raptorxml exportresourcestrings de c:\Strings.xml
raptorxmlserver exportresourcestrings de c:\Strings.xml
```

- El primer comando crea un archivo llamado Strings.xml en la carpeta c:\; ese archivo contiene las cadenas de los recursos de RaptorXML Server en inglés.
- El segundo comando llama al ejecutable del servidor y hace lo mismo que el primero.

Localizar RaptorXML Server en otros idiomas

Si quiere puede localizar RaptorXML Server en cualquier idioma. Altova ya ofrece la aplicación en cinco idiomas: inglés, español, francés, alemán y japonés (todos los archivos están en la carpeta C:\Program Files (x86)\Altova\RaptorXMLServer2024\bin) pero puede localizarla en cualquier otro idioma.

Siga estos pasos para localizar la aplicación:

1. Genere un archivo XML con las cadenas de recursos usando el comando exportresourcestrings (*ver más arriba*). Las cadenas de recursos de este archivo XML puede estar en uno de estos idiomas: inglés (en), español (es), francés (fr), alemán (de) o japonés (ja), dependiendo del argumento ***CódigoIdioma*** que utilice con el comando.
2. Traduzca las cadenas de recursos al idioma de destino. Las cadenas de recursos son el contenido de los elementos <string> del archivo XML. No traduzca las variables que aparecen entre llaves, p. ej. {option} o {product}.
3. Póngase en contacto con el equipo de soporte técnico de Altova, que le ayudarán a generar un archivo DLL localizado de RaptorXML Server a partir de su archivo XML traducido.
4. Cuando reciba el archivo DLL localizado del equipo de soporte técnico de Altova, guárdelo en la

 © 2017-2023 Altova GmbH

carpeta `C:\Archivos de programas (x86)\Altova\RaptorXMLServer2024\bin`. El DLL tendrá un nombre similar a este `RaptorXML2024_ci.dll`. La parte `_ci` del nombre contiene el código del idioma. Por ejemplo, en `RaptorXML2024_de.dll`, la parte `de` es el código del idioma alemán (Deutsch).

1. Ejecute el comando `setdeflang` para establecer el archivo DLL localizado como aplicación RaptorXML Server predeterminada. Use el código de idioma del nombre del archivo DLL como argumento del comando `setdeflang`

Nota: Altova ya ofrece RaptorXML Server en estos cinco idiomas: inglés, español, francés, alemán y japonés. Para usar uno de estos idiomas como idioma predeterminado, use el comando `setdeflang` de RaptorXML Server.

5.8.2 setdeflang

Sintaxis y descripción

El comando **`setdeflang`** (**`sdl`** en versión corta) establece el idioma predeterminado de RaptorXML Server. Los idiomas disponibles son inglés (`en`), alemán (`de`), español (`es`), francés (`fr`) y japonés (`ja`). El comando toma el argumento obligatorio *`LanguageCode`*.

```
raptorxml setdeflang [opciones] LanguageCode
raptorxmlserver setdeflang [opciones] LanguageCode
```

- El argumento *`LanguageCode`* es obligatorio y define el idioma predeterminado de RaptorXML Server. Los valores correspondientes son: `en`, `de`, `es`, `fr`, `ja`.
- Puede llamar al comando **`setdeflang`** desde estos ejecutables: **`raptorxml`** o **`raptorxmlserver`**.
- Use la opción **`--h, --help`** para mostrar información sobre el comando.

▼ Mayúsculas/minúsculas y barras en la línea de comandos

`RaptorXML` (y **`RaptorXMLServer`** para comandos administrativos) *en Windows*
`raptorxml` (y **`raptorxmlserver`** para comandos administrativos) *en Windows y Unix (Linux, Mac)*

* Las minúsculas (`raptorxml` y `raptorxmlserver`) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que las mayúsculas (`RaptorXML`) funcionan solamente en Windows y Mac.
* Use la barra diagonal en Linux y Mac y la barra diagonal inversa en Windows.

Ejemplo

Ejemplos del comando **`setdeflang`** (**`sdl`**):

```
raptorxml sdl de
raptorxml setdeflang es
raptorxmlserver setdeflang es
```

- El primer comando define el alemán como idioma predeterminado de RaptorXML Server.
- El segundo comando define el español como idioma predeterminado de RaptorXML Server.
- El tercer comando es el mismo que el segundo, pero lo ejecuta el ejecutable del servidor.

Opciones

Use la opción `--h, --help` para mostrar información sobre el comando.

© 2017-2023 Altova GmbH

5.9 Comandos de licencias

En este apartado puede consultar las descripciones de los comandos que puede usar para asignar una licencia a RaptorXML Server:

licenseserver 217	Registra RaptorXML Server con Altova LicenseServer en su red.
assignlicense 219	Carga un archivo de licencia en LicenseServer (solo en Windows).
verifylicense 220	Comprueba si RaptorXML Server tiene asignada una licencia (solo en Windows).

Nota: estos comandos también se pueden ejecutar con el ejecutable del servidor para los comandos de gestión 222.

Para más información sobre cómo asignar licencias a productos de Altova con Altova LicenseServer consulte la documentación de Altova LicenseServer.

5.9.1 licenseserver

Sintaxis y descripción

Al ejecutarse, el comando `licenseserver` registra RaptorXML Server con el servidor LicenseServer indicado por el argumento *`Servidor-O-Dirección-IP`*. Para que el comando `licenseserver` se ejecute correctamente, los dos servidores (RaptorXML Server y LicenseServer) deben estar conectados en la red y LicenseServer debe estar en ejecución. Además debe tener privilegios de administrador para poder registrar RaptorXML Server con LicenseServer.

```
raptorxml licenseserver [opciones] Server-Or-IP-Address
raptorxmlserver licenseserver [opciones] Server-Or-IP-Address
```

- El argumento *`Server-Or-IP-Address`* toma el nombre o la dirección IP del equipo en el que se está ejecutando LicenseServer.
- Para solicitar una licencia de evaluación para RaptorXML Server, indique los valores correspondientes para las opciones `--company`, `--email` y `--name`. Para solicitar una licencia de evaluación para otro producto de Altova (además de RaptorXML Server) agregue la opción `--request-eval-license` y use como valor el nombre del producto para el que quiere la licencia. Consulte más abajo la lista de opciones para más información.
- Puede llamar al comando **`licenseserver`** desde estos dos ejecutables: **`raptorxml`** o **`raptorxmlserver`**.

Cuando RaptorXML Server se registre con LicenseServer, recibirá un mensaje de confirmación. El mensaje incluirá la URL del servidor LicenseServer. Ahora puede usar la URL para ir a LicenseServer y asignarle una licencia a RaptorXML Server. Consulte la documentación de Altova LicenseServer para obtener más información (https://www.altova.com/manual/es/licenseserver/3.12/).

▼ Mayúsculas/minúsculas y barras en la línea de comandos

RaptorXML (y **RaptorXMLServer** para comandos administrativos) *en Windows*
raptorxml (y **raptorxmlserver** para comandos administrativos) *en Windows y Unix (Linux, Mac)*

* Las minúsculas (raptorxml y raptorxmlserver) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que las mayúsculas (RaptorXML) funcionan solamente en Windows y Mac.
* Use la barra diagonal en Linux y Mac y la barra diagonal inversa en Windows.

- Barra diagonal inversa y espacios en sistemas Windows

  En sistemas Windows: si hay espacios o caracteres especiales (por ejemplo en los nombres de archivos o de carpetas, empresas, personas o productos) debe usar comillas: por ejemplo, **"Mi archivo"**. Sin embargo, debe tener en cuenta que una barra diagonal inversa seguida de comillas dobles (por ejemplo, **"C:\Mi Directorio\"**) es posible que no se lea correctamente. Esto se debe a que la barra diagonal inversa también se usa para empezar secuencias de escape y la secuencia de escape **\"** también usa las comillas dobles. Para evitar secuencia de caracteres puede añadir otra barra diagonal inversa: **\\"**. En resumen: si necesita escribir una ruta que contenga espacios y una barra diagonal inversa, esta es la mejor forma de hacerlo: **"C:\Mi Directorio\\"**.

Ejemplo

Ejemplo del comando **licenseserver**:

```
raptorxml licenseserver DOC.altova.com
raptorxml licenseserver localhost
raptorxml licenseserver 127.0.0.1
raptorxmlserver licenseserver 127.0.0.1
```

Estos comandos indican respectivamente que el equipo que ejecuta el servidor Altova LicenseServer es un equipo llamado DOC.altova.com y el equipo del usuario (localhost y 127.0.0.1). En cada caso el comando registra RaptorXML Server con el servidor LicenseServer del equipo correspondiente. El último comando indica al ejecutable del servidor que ejecute el comando.

Opciones

Las opciones se enumeran en versión corta (si existe) y larga. Puede usar una o dos barras tanto para la versión corta como para la larga. Las opciones pueden tomar un valor o no hacerlo. Si lo hacen, se escribe así: **--option=value**. Los valores se pueden indicar sin comillas, salvo en dos casos: (i) cuando la cadena del valor contiene espacios o (ii) cuando se indica de forma explícita en la descripción de la opción que las comillas son necesarias. Si una opción toma un valor booleano y no se especifica ningún valor, entonces el valor predeterminado de la opción es TRUE. Use la opción **--h, --help** para mostrar información sobre el comando.

- json [j]

  --j, --json = true|false
  Usa los valores true|false. Si se establece en true imprime el resultado del intento de registro como objeto JSON analizable por máquina

 © 2017-2023 Altova GmbH

5.9.2 assignlicense (solo Windows)

Sintaxis y descripción

El comando `assignlicense` está disponible en Windows solamente (no en sistemas Linux o Mac). Al ejecutarse carga el archivo de licencia indicado por el argumento *ARCHIVO* en el servidor LicenseServer con el que está registrado RaptorXML Server (véase el comando `licenseserver`) y asigna esa licencia a RaptorXML Server en este equipo. El argumento *ARCHIVO* toma la ruta de acceso del archivo de licencia. El comando también permite comprobar la validez de las licencias.

```
raptorxml assignlicense [opciones] FILE
raptorxmlserver assignlicense [opciones] FILE
```

- El argumento *FILE* toma la ruta al archivo de la licencia.
- La opción `--test-only` sirve para cargar la licencia a LicenseServer y validarla sin asignarla primero a RaptorXML Server.
- El comando `assignlicense` se puede llamar desde estos dos ejecutables: **raptorxml** o **raptorxmlserver**.

Para más información sobre el proceso de asignación de licencias consulte la documentación de Altova LicenseServer (https://www.altova.com/manual/es/licenseserver/3.12/).

▼ Mayúsculas/minúsculas y barras en la línea de comandos

RaptorXML (y **RaptorXMLServer** para comandos administrativos) *en Windows*
raptorxml (y **raptorxmlserver** para comandos administrativos) *en Windows y Unix (Linux, Mac)*

* Las minúsculas (`raptorxml` y `raptorxmlserver`) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que las mayúsculas (`RaptorXML`) funcionan solamente en Windows y Mac.
* Use la barra diagonal en Linux y Mac y la barra diagonal inversa en Windows.

▼ Barra diagonal inversa y espacios en sistemas Windows

En sistemas Windows: si hay espacios o caracteres especiales (por ejemplo en los nombres de archivos o de carpetas, empresas, personas o productos) debe usar comillas: por ejemplo, **"Mi archivo"**. Sin embargo, debe tener en cuenta que una barra diagonal inversa seguida de comillas dobles (por ejemplo, **"C:\Mi Directorio\"**) es posible que no se lea correctamente. Esto se debe a que la barra diagonal inversa también se usa para empezar secuencias de escape y la secuencia de escape **\"** también usa las comillas dobles. Para evitar secuencia de caracteres puede añadir otra barra diagonal inversa: **\\"**. En resumen: si necesita escribir una ruta que contenga espacios y una barra diagonal inversa, esta es la mejor forma de hacerlo: **"C:\Mi Directorio\\"**.

Ejemplos

Ejemplos del comando `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
```

- El primer ejemplo carga la licencia indicada a LicenseServer y se la asigna a RaptorXML Server.
- El segundo ejemplo carga la licencia indicada a LicenseServer y la valida, sin asignársela a RaptorXML Server.
- El tercer comando carga la licencia indicada a LicenseServer y la valida, sin asignársela a RaptorXML Server.

Opciones

Las opciones se enumeran en versión corta (si existe) y larga. Puede usar una o dos barras tanto para la versión corta como para la larga. Las opciones pueden tomar un valor o no hacerlo. Si lo hacen, se escribe así: **`--option=value`**. Los valores se pueden indicar sin comillas, salvo en dos casos: (i) cuando la cadena del valor contiene espacios o (ii) cuando se indica de forma explícita en la descripción de la opción que las comillas son necesarias. Si una opción toma un valor booleano y no se especifica ningún valor, entonces el valor predeterminado de la opción es `TRUE`. Use la opción **`--h, --help`** para mostrar información sobre el comando.

▼ test-only [t]

`--t, --test-only =` `true|false`
Son valores permitidos: `true|false`. Si su valor es `true`, el archivo de licencia solamente se carga y valida en LicenseServer, pero no se asigna

5.9.3 verifylicense (solo Windows)

Sintaxis y descripción

El comando **`verifylicense`** comprueba si un producto tiene licencia. La opción `--license-key` también permite comprobar si ya se ha asignado al producto una clave de licencia. Este comando solo funciona en Windows. No se puede usar en sistemas Linux ni Mac.

```
raptorxml verifylicense [opciones]
raptorxmlserver verifylicense [opciones]
```

- Para comprobar si una licencia en concreto está asignada a RaptorXML Server debe indicar la clave de licencia como valor de la opción `--license-key`.
- Puede llamar al comando **`verifylicense`** desde estos ejecutables: **`raptorxml`** o **`raptorxmlserver`**.

Para más información sobre el proceso de asignación de licencias consulte la documentación de LicenseServer (https://www.altova.com/manual/es/licenseserver/3.12/).

▼ Mayúsculas/minúsculas y barras en la línea de comandos

`RaptorXML` (y **`RaptorXMLServer`** para comandos administrativos) *en Windows*
`raptorxml` (y **`raptorxmlserver`** para comandos administrativos) *en Windows y Unix (Linux, Mac)*

* Las minúsculas (`raptorxml` y `raptorxmlserver`) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que las mayúsculas (`RaptorXML`) funcionan solamente en Windows y Mac.
* Use la barra diagonal en Linux y Mac y la barra diagonal inversa en Windows.

 © 2017-2023 Altova GmbH

Ejemplo

Ejemplo del comando **verifylicense**:

```
raptorxml verifylicense
raptorxml verifylicense --license-key=ABCD123-ABCD123-ABCD123-ABCD123-ABCD123-ABCD123
raptorxmlserver verifylicense --license-key=ABCD123-ABCD123-ABCD123-ABCD123-ABCD123-
ABCD123
```

- El primer comando comprueba si RaptorXML Server tiene licencia.
- El segundo comando comprueba si RaptorXML Server usa la clave de licencia indicada en la opción `--license-key`.
- El tercer comando es el mismo que el segundo pero lo ejecuta el ejecutable del servidor.

Opciones

Las opciones se enumeran en versión corta (si existe) y larga. Puede usar una o dos barras tanto para la versión corta como para la larga. Las opciones pueden tomar un valor o no hacerlo. Si lo hacen, se escribe así: **--option=value**. Los valores se pueden indicar sin comillas, salvo en dos casos: (i) cuando la cadena del valor contiene espacios o (ii) cuando se indica de forma explícita en la descripción de la opción que las comillas son necesarias. Si una opción toma un valor booleano y no se especifica ningún valor, entonces el valor predeterminado de la opción es `TRUE`. Use la opción **--h, --help** para mostrar información sobre el comando.

▼ license-key [l]

--l, --license-key = *Value*
Comprueba si RaptorXML Server usa la clave de licencia indicada como valor de esta opción.

5.10 Comandos de gestión

Los comandos de gestión (como los comandos de instalación como servicio o de licencias) se emiten al ejecutable del servidor de RaptorXML Server (llamado **RaptorXMLServer**). Este ejecutable se encuentra por defecto en:

Windows	`<ProgramFilesFolder>\Altova\RaptorXMLServer2024\bin\`**`RaptorXMLServer.exe`**
Linux	`/opt/Altova/RaptorXMLServer2024/bin/`**`raptorxmlserver`**
Mac	`/usr/local/Altova/RaptorXMLServer2024/bin/`**`raptorxmlserver`**

Uso

La sintaxis de la línea de comandos es:

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

- `--help` (`--h` en versión abreviada): muestra el texto de ayuda del comando que se indique. Si no se especifica ningún comando, entonces aparecen todos los comandos del ejecutable, cada uno con una breve descripción.
- `--version`: muestra el número de versión de RaptorXML Server.
- `<command>`: es el comando que se quiere ejecutar. Los comandos aparecen en los subapartados de este apartado (*véase la lista más abajo*).
- `[options]`: las opciones de un comando; cada comando aparece con su descripción y comandos correspondientes.
- `[arguments]`: los argumentos de un comando; cada comando aparece con una lista de sus argumentos correspondientes.

▼ Mayúsculas/minúsculas y barras en la línea de comandos

RaptorXML (y **RaptorXMLServer** para comandos administrativos) *en Windows*
raptorxml (y **raptorxmlserver** para comandos administrativos) *en Windows y Unix (Linux, Mac)*

* Las minúsculas (`raptorxml` y `raptorxmlserver`) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que las mayúsculas (`RaptorXML`) funcionan solamente en Windows y Mac.
* Use la barra diagonal en Linux y Mac y la barra diagonal inversa en Windows.

Comandos de gestión

Los comandos del ejecutable del servidor permiten llamar a las funciones de gestión, que se describen en este apartado y que son:

- install 223
- uninstall 223
- start 224
- setdeflang 225
- licenseserver 226
- assignlicense 228

 © 2017-2023 Altova GmbH

- verifylicense 229
- createconfig 230
- exportresourcestrings 231
- debug 233
- help 234

5.10.1 install

Sintaxis y descripción

El comando `install` instala RaptorXML Server como servicio en el equipo servidor.

```
raptorxmlserver install [opciones]
```

- Al instalar RaptorXML Server como un servicio el servicio no se inicia automáticamente. Para iniciar el servicio use el comando **`start`**.
- Para desinstalar RaptorXML Server como servicio, use el comando **`uninstall`**.
- Use la opción **`--h, --help`** para mostrar información sobre el comando.

▼ Mayúsculas/minúsculas y barras en la línea de comandos

`RaptorXML` (y **`RaptorXMLServer`** para comandos administrativos) *en Windows*
`raptorxml` (y **`raptorxmlserver`** para comandos administrativos) *en Windows y Unix (Linux, Mac)*

* Las minúsculas (`raptorxml` y `raptorxmlserver`) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que las mayúsculas (`RaptorXML`) funcionan solamente en Windows y Mac.
* Use la barra diagonal en Linux y Mac y la barra diagonal inversa en Windows.

Ejemplo

Ejemplo del comando **`install`**:

```
raptorxmlserver install
```

5.10.2 uninstall

Sintaxis y descripción

El comando **`uninstall`** desinstala RaptorXML Server como servicio en el equipo servidor.

```
raptorxmlserver uninstall [opciones]
```

Para volver a instalar RaptorXML Server como servicio use el comando **`install`**.

▼ Mayúsculas/minúsculas y barras en la línea de comandos

`RaptorXML` (y **`RaptorXMLServer`** para comandos administrativos) *en Windows*
`raptorxml` (y **`raptorxmlserver`** para comandos administrativos) *en Windows y Unix (Linux, Mac)*

* Las minúsculas (`raptorxml` y `raptorxmlserver`) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que las mayúsculas (`RaptorXML`) funcionan solamente en Windows y Mac.
* Use la barra diagonal en Linux y Mac y la barra diagonal inversa en Windows.

Ejemplo

Ejemplo del comando **`uninstall`**:

```
raptorxmlserver uninstall
```

5.10.3 start

Sintaxis y descripción

El comando `start` inicia RaptorXML Server como servicio en el equipo servidor.

```
raptorxmlserver start [opciones]
```

- Si RaptorXML Server no está instalado como servicio, puede hacerlo con el comando `install` (antes de iniciarlo).
- Para desinstalar RaptorXML Server como servicio use el comando **`uninstall`**.
- Use la opción **`--h, --help`** para ver información sobre el comando.

▼ Mayúsculas/minúsculas y barras en la línea de comandos

`RaptorXML` (y **`RaptorXMLServer`** para comandos administrativos) *en Windows*
`raptorxml` (y **`raptorxmlserver`** para comandos administrativos) *en Windows y Unix (Linux, Mac)*

* Las minúsculas (`raptorxml` y `raptorxmlserver`) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que las mayúsculas (`RaptorXML`) funcionan solamente en Windows y Mac.
* Use la barra diagonal en Linux y Mac y la barra diagonal inversa en Windows.

▼ Barra diagonal inversa y espacios en sistemas Windows

En sistemas Windows: si hay espacios o caracteres especiales (por ejemplo en los nombres de archivos o de carpetas, empresas, personas o productos) debe usar comillas: por ejemplo, **`"Mi archivo"`**. Sin embargo, debe tener en cuenta que una barra diagonal inversa seguida de comillas dobles (por ejemplo, **`"C:\Mi Directorio\"`**) es posible que no se lea correctamente. Esto se debe a que la barra diagonal inversa también se usa para empezar secuencias de escape y la secuencia de escape **`\"`** también usa las comillas dobles. Para evitar secuencia de caracteres puede añadir otra barra diagonal inversa: **`\\"`**. En resumen: si necesita escribir una ruta que contenga espacios y una barra diagonal inversa, esta es la mejor forma de hacerlo: **`"C:\Mi Directorio\\"`**.

Ejemplo

Ejemplo del comando **`start`**:

 © 2017-2023 Altova GmbH

```
raptorxmlserver start
```

Opciones

Las opciones se enumeran en versión corta (si existe) y larga. Puede usar una o dos barras tanto para la versión corta como para la larga. Las opciones pueden tomar un valor o no hacerlo. Si lo hacen, se escribe así: **--option=value**. Los valores se pueden indicar sin comillas, salvo en dos casos: (i) cuando la cadena del valor contiene espacios o (ii) cuando se indica de forma explícita en la descripción de la opción que las comillas son necesarias. Si una opción toma un valor booleano y no se especifica ningún valor, entonces el valor predeterminado de la opción es `TRUE`. Use la opción **--h, --help** para mostrar información sobre el comando.

- config [c]

  --c, --config = *File*
  Indica la ruta de acceso a un archivo de configuración.

- fork

  --fork = `true|false`
  Permite bifurcar al usar `init` en servidores Unix. El valor predeterminado es `false`.

- port

  --port = *PortNumber*
  El número de puerto de la instancia de depuración de RaptorXML Server.

5.10.4 setdeflang

Sintaxis y descripción

El comando **setdeflang** (**sdl** en versión corta) establece el idioma predeterminado de RaptorXML Server. Los idiomas disponibles son inglés (`en`), alemán (`de`), español (`es`), francés (`fr`) y japonés (`ja`). El comando toma el argumento obligatorio *LanguageCode*.

```
raptorxml setdeflang [opciones] LanguageCode
raptorxmlserver setdeflang [opciones] LanguageCode
```

- El argumento *LanguageCode* es obligatorio y define el idioma predeterminado de RaptorXML Server. Los valores correspondientes son: `en`, `de`, `es`, `fr`, `ja`.
- Puede llamar al comando **setdeflang** desde estos ejecutables: **raptorxml** o **raptorxmlserver**.
- Use la opción **--h, --help** para mostrar información sobre el comando.

- Mayúsculas/minúsculas y barras en la línea de comandos

  RaptorXML (y **RaptorXMLServer** para comandos administrativos) *en Windows*
  raptorxml (y **raptorxmlserver** para comandos administrativos) *en Windows y Unix (Linux, Mac)*

  * Las minúsculas (`raptorxml` y `raptorxmlserver`) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que las mayúsculas (`RaptorXML`) funcionan solamente en Windows y Mac.

* Use la barra diagonal en Linux y Mac y la barra diagonal inversa en Windows.

Ejemplo

Ejemplos del comando **setdeflang** (**sdl**):

```
raptorxml sdl de
raptorxml setdeflang es
raptorxmlserver setdeflang es
```

- El primer comando define el alemán como idioma predeterminado de RaptorXML Server.
- El segundo comando define el español como idioma predeterminado de RaptorXML Server.
- El tercer comando es el mismo que el segundo, pero lo ejecuta el ejecutable del servidor.

Opciones

Use la opción **-h, --help** para mostrar información sobre el comando.

5.10.5 licenseserver

Sintaxis y descripción

Al ejecutarse, el comando `licenseserver` registra RaptorXML Server con el servidor LicenseServer indicado por el argumento *Servidor-O-Dirección-IP*. Para que el comando `licenseserver` se ejecute correctamente, los dos servidores (RaptorXML Server y LicenseServer) deben estar conectados en la red y LicenseServer debe estar en ejecución. Además debe tener privilegios de administrador para poder registrar RaptorXML Server con LicenseServer.

```
raptorxml licenseserver [opciones] Server-Or-IP-Address
raptorxmlserver licenseserver [opciones] Server-Or-IP-Address
```

- El argumento *Server-Or-IP-Address* toma el nombre o la dirección IP del equipo en el que se está ejecutando LicenseServer.
- Para solicitar una licencia de evaluación para RaptorXML Server, indique los valores correspondientes para las opciones `--company`, `--email` y `--name`. Para solicitar una licencia de evaluación para otro producto de Altova (además de RaptorXML Server) agregue la opción `--request-eval-license` y use como valor el nombre del producto para el que quiere la licencia. Consulte más abajo la lista de opciones para más información.
- Puede llamar al comando **licenseserver** desde estos dos ejecutables: **raptorxml** o **raptorxmlserver**.

Cuando RaptorXML Server se registre con LicenseServer, recibirá un mensaje de confirmación. El mensaje incluirá la URL del servidor LicenseServer. Ahora puede usar la URL para ir a LicenseServer y asignarle una licencia a RaptorXML Server. Consulte la documentación de Altova LicenseServer para obtener más información (https://www.altova.com/manual/es/licenseserver/3.12/).

▼ Mayúsculas/minúsculas y barras en la línea de comandos

 © 2017-2023 Altova GmbH

RaptorXML (y **RaptorXMLServer** para comandos administrativos) *en Windows*
raptorxml (y **raptorxmlserver** para comandos administrativos) *en Windows y Unix (Linux, Mac)*

* Las minúsculas (raptorxml y raptorxmlserver) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que las mayúsculas (RaptorXML) funcionan solamente en Windows y Mac.
* Use la barra diagonal en Linux y Mac y la barra diagonal inversa en Windows.

- Barra diagonal inversa y espacios en sistemas Windows

  En sistemas Windows: si hay espacios o caracteres especiales (por ejemplo en los nombres de archivos o de carpetas, empresas, personas o productos) debe usar comillas: por ejemplo, **"Mi archivo"**. Sin embargo, debe tener en cuenta que una barra diagonal inversa seguida de comillas dobles (por ejemplo, **"C:\Mi Directorio\"**) es posible que no se lea correctamente. Esto se debe a que la barra diagonal inversa también se usa para empezar secuencias de escape y la secuencia de escape **\"** también usa las comillas dobles. Para evitar secuencia de caracteres puede añadir otra barra diagonal inversa: **\\"**. En resumen: si necesita escribir una ruta que contenga espacios y una barra diagonal inversa, esta es la mejor forma de hacerlo: **"C:\Mi Directorio\\"**.

Ejemplo

Ejemplo del comando **licenseserver**:

```
raptorxml licenseserver DOC.altova.com
raptorxml licenseserver localhost
raptorxml licenseserver 127.0.0.1
raptorxmlserver licenseserver 127.0.0.1
```

Estos comandos indican respectivamente que el equipo que ejecuta el servidor Altova LicenseServer es un equipo llamado DOC.altova.com y el equipo del usuario (localhost y 127.0.0.1). En cada caso el comando registra RaptorXML Server con el servidor LicenseServer del equipo correspondiente. El último comando indica al ejecutable del servidor que ejecute el comando.

Opciones

Las opciones se enumeran en versión corta (si existe) y larga. Puede usar una o dos barras tanto para la versión corta como para la larga. Las opciones pueden tomar un valor o no hacerlo. Si lo hacen, se escribe así: **--option=value**. Los valores se pueden indicar sin comillas, salvo en dos casos: (i) cuando la cadena del valor contiene espacios o (ii) cuando se indica de forma explícita en la descripción de la opción que las comillas son necesarias. Si una opción toma un valor booleano y no se especifica ningún valor, entonces el valor predeterminado de la opción es TRUE. Use la opción **--h, --help** para mostrar información sobre el comando.

- json [j]

  --j, --json = true|false
  Usa los valores true|false. Si se establece en true imprime el resultado del intento de registro como objeto JSON analizable por máquina

5.10.6 assignlicense (Windows only)

Sintaxis y descripción

El comando `assignlicense` está disponible en Windows solamente (no en sistemas Linux o Mac). Al ejecutarse carga el archivo de licencia indicado por el argumento *ARCHIVO* en el servidor LicenseServer con el que está registrado RaptorXML Server (véase el comando `licenseserver`) y asigna esa licencia a RaptorXML Server en este equipo. El argumento *ARCHIVO* toma la ruta de acceso del archivo de licencia. El comando también permite comprobar la validez de las licencias.

```
raptorxml assignlicense [opciones] FILE
raptorxmlserver assignlicense [opciones] FILE
```

- El argumento *FILE* toma la ruta al archivo de la licencia.
- La opción `--test-only` sirve para cargar la licencia a LicenseServer y validarla sin asignarla primero a RaptorXML Server.
- El comando `assignlicense` se puede llamar desde estos dos ejecutables: **raptorxml** o **raptorxmlserver**.

Para más información sobre el proceso de asignación de licencias consulte la documentación de Altova LicenseServer (https://www.altova.com/manual/es/licenseserver/3.12/).

▼ Mayúsculas/minúsculas y barras en la línea de comandos

RaptorXML (y **RaptorXMLServer** para comandos administrativos) *en Windows*
raptorxml (y **raptorxmlserver** para comandos administrativos) *en Windows y Unix (Linux, Mac)*

* Las minúsculas (`raptorxml` y `raptorxmlserver`) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que las mayúsculas (`RaptorXML`) funcionan solamente en Windows y Mac.
* Use la barra diagonal en Linux y Mac y la barra diagonal inversa en Windows.

▼ Barra diagonal inversa y espacios en sistemas Windows

En sistemas Windows: si hay espacios o caracteres especiales (por ejemplo en los nombres de archivos o de carpetas, empresas, personas o productos) debe usar comillas: por ejemplo, **"Mi archivo"**. Sin embargo, debe tener en cuenta que una barra diagonal inversa seguida de comillas dobles (por ejemplo, **"C:\Mi Directorio\"**) es posible que no se lea correctamente. Esto se debe a que la barra diagonal inversa también se usa para empezar secuencias de escape y la secuencia de escape **\"** también usa las comillas dobles. Para evitar secuencia de caracteres puede añadir otra barra diagonal inversa: **\\"**. En resumen: si necesita escribir una ruta que contenga espacios y una barra diagonal inversa, esta es la mejor forma de hacerlo: **"C:\Mi Directorio\\"**.

Ejemplos

Ejemplos del comando `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
```

 © 2017-2023 Altova GmbH

- El primer ejemplo carga la licencia indicada a LicenseServer y se la asigna a RaptorXML Server.
- El segundo ejemplo carga la licencia indicada a LicenseServer y la valida, sin asignársela a RaptorXML Server.
- El tercer comando carga la licencia indicada a LicenseServer y la valida, sin asignársela a RaptorXML Server.

Opciones

Las opciones se enumeran en versión corta (si existe) y larga. Puede usar una o dos barras tanto para la versión corta como para la larga. Las opciones pueden tomar un valor o no hacerlo. Si lo hacen, se escribe así: **`--option=value`**. Los valores se pueden indicar sin comillas, salvo en dos casos: (i) cuando la cadena del valor contiene espacios o (ii) cuando se indica de forma explícita en la descripción de la opción que las comillas son necesarias. Si una opción toma un valor booleano y no se especifica ningún valor, entonces el valor predeterminado de la opción es `TRUE`. Use la opción **`--h, --help`** para mostrar información sobre el comando.

▼ test-only [t]

`--t, --test-only =` `true|false`
Son valores permitidos: `true|false`. Si su valor es `true`, el archivo de licencia solamente se carga y valida en LicenseServer, pero no se asigna

5.10.7 verifylicense (Windows only)

Sintaxis y descripción

El comando **`verifylicense`** comprueba si un producto tiene licencia. La opción `--license-key` también permite comprobar si ya se ha asignado al producto una clave de licencia. Este comando solo funciona en Windows. No se puede usar en sistemas Linux ni Mac.

```
raptorxml verifylicense [opciones]
raptorxmlserver verifylicense [opciones]
```

- Para comprobar si una licencia en concreto está asignada a RaptorXML Server debe indicar la clave de licencia como valor de la opción `--license-key`.
- Puede llamar al comando **`verifylicense`** desde estos ejecutables: **`raptorxml`** o **`raptorxmlserver`**.

Para más información sobre el proceso de asignación de licencias consulte la documentación de LicenseServer (https://www.altova.com/manual/es/licenseserver/3.12/).

▼ Mayúsculas/minúsculas y barras en la línea de comandos

`RaptorXML` (y **`RaptorXMLServer`** para comandos administrativos) *en Windows*
`raptorxml` (y **`raptorxmlserver`** para comandos administrativos) *en Windows y Unix (Linux, Mac)*

* Las minúsculas (`raptorxml` y `raptorxmlserver`) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que las mayúsculas (`RaptorXML`) funcionan solamente en Windows y Mac.
* Use la barra diagonal en Linux y Mac y la barra diagonal inversa en Windows.

Ejemplo

Ejemplo del comando **verifylicense**:

```
raptorxml verifylicense
raptorxml verifylicense --license-key=ABCD123-ABCD123-ABCD123-ABCD123-ABCD123-ABCD123
raptorxmlserver verifylicense --license-key=ABCD123-ABCD123-ABCD123-ABCD123-ABCD123-
ABCD123
```

- El primer comando comprueba si RaptorXML Server tiene licencia.
- El segundo comando comprueba si RaptorXML Server usa la clave de licencia indicada en la opción `--license-key`.
- El tercer comando es el mismo que el segundo pero lo ejecuta el ejecutable del servidor.

Opciones

Las opciones se enumeran en versión corta (si existe) y larga. Puede usar una o dos barras tanto para la versión corta como para la larga. Las opciones pueden tomar un valor o no hacerlo. Si lo hacen, se escribe así: **--option=value**. Los valores se pueden indicar sin comillas, salvo en dos casos: (i) cuando la cadena del valor contiene espacios o (ii) cuando se indica de forma explícita en la descripción de la opción que las comillas son necesarias. Si una opción toma un valor booleano y no se especifica ningún valor, entonces el valor predeterminado de la opción es `TRUE`. Use la opción **--h, --help** para mostrar información sobre el comando.

▼ license-key [l]

--l, --license-key = *Value*
Comprueba si RaptorXML Server usa la clave de licencia indicada como valor de esta opción.

5.10.8 createconfig

Sintaxis y descripción

El comando **createconfig** sobrescribe el archivo de configuración del servidor con valores predeterminados.

```
raptorxmlserver createconfig [options]
```

- La opción **--lang** indica el idioma predeterminado del archivo de configuración del servidor.

Para más información sobre los archivos de configuración del servidor consulte el apartado Configurar el servidor 258.

▼ Casing and slashes on the command line

RaptorXML (and **RaptorXMLServer** for administration commands) *on Windows*
raptorxml (and **raptorxmlserver** for administration commands) *on Windows and Unix (Linux, Mac)*

* Note that lowercase (`raptorxml` and `raptorxmlserver`) works on all platforms (Windows, Linux, and Mac), while upper-lower (`RaptorXML`) works only on Windows and Mac.

 © 2017-2023 Altova GmbH

* Use forward slashes on Linux and Mac, backslashes on Windows.

Ejemplos

Ejemplos del comando **createconfig**:

```
raptorxml createconfig
raptorxml createconfig --lang=de
```

Opciones

- lang

  --lang = en|de|es|fr|ja
  Indica el idioma predeterminado del archivo de configuración del servidor. Se pueden usar estas opciones: inglés (en), alemán (de), español (es), francés (fr) y japonés (ja). Si no se especifica esta opción el idioma predeterminado es el inglés.

Use la opción **--h, --help** para mostrar información sobre el comando.

Las opciones se enumeran en versión corta (si existe) y larga. Puede usar una o dos barras tanto para la versión corta como para la larga. Las opciones pueden tomar un valor o no hacerlo. Si lo hacen, se escribe así: **--option=value**. Los valores se pueden indicar sin comillas, salvo en dos casos: (i) cuando la cadena del valor contiene espacios o (ii) cuando se indica de forma explícita en la descripción de la opción que las comillas son necesarias. Si una opción toma un valor booleano y no se especifica ningún valor, entonces el valor predeterminado de la opción es TRUE. Use la opción **--h, --help** para mostrar información sobre el comando.

5.10.9 exportresourcestrings

Sintaxis y descripción

El comando exportresourcestrings genera un archivo XML que contiene todas las cadenas de recursos de la aplicación RaptorXML Server en el idioma indicado. Los idiomas en los que se puede generar el archivo son inglés (en), español (es), francés (fr) alemán (de) y japonés (ja).

```
raptorxml exportresourcestrings [opciones] LanguageCode XMLOutputFile
raptorxmlserver exportresourcestrings [opciones] LanguageCode XMLOutputFile
```

- El argumento ***LanguageCode*** indica el idioma de las cadenas de recursos del archivo XML de salida; se trata del *lenguaje de exportación*. Se admiten estos idiomas de exportación (se indica el código correspondiente entre paréntesis): inglés (en), alemán, (de), español (es), francés (fr) y japonés (ja).
- El argumento ***XMLOutputFile*** indica la ruta y el nombre del archivo XML de salida.
- Puede llamar al comando **exportresourcestrings** desde desde estos dos ejecutables: **raptorxml** o **raptorxmlserver**.

A continuación explicamos cómo localizar.

▼ Mayúsculas/minúsculas y barras en la línea de comandos

RaptorXML (y **RaptorXMLServer** para comandos administrativos) *en Windows*
raptorxml (y **raptorxmlserver** para comandos administrativos) *en Windows y Unix (Linux, Mac)*

* Las minúsculas (raptorxml y raptorxmlserver) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que las mayúsculas (RaptorXML) funcionan solamente en Windows y Mac.
* Use la barra diagonal en Linux y Mac y la barra diagonal inversa en Windows.

▼ Barra diagonal inversa y espacios en sistemas Windows

En sistemas Windows: si hay espacios o caracteres especiales (por ejemplo en los nombres de archivos o de carpetas, empresas, personas o productos) debe usar comillas: por ejemplo, **"Mi archivo"**. Sin embargo, debe tener en cuenta que una barra diagonal inversa seguida de comillas dobles (por ejemplo, **"C:\Mi Directorio\"**) es posible que no se lea correctamente. Esto se debe a que la barra diagonal inversa también se usa para empezar secuencias de escape y la secuencia de escape **\"** también usa las comillas dobles. Para evitar secuencia de caracteres puede añadir otra barra diagonal inversa: **\\"**. En resumen: si necesita escribir una ruta que contenga espacios y una barra diagonal inversa, esta es la mejor forma de hacerlo: **"C:\Mi Directorio\\"**.

Ejemplos

Ejemplos del comando exportresourcestrings:

```
raptorxml exportresourcestrings de c:\Strings.xml
raptorxmlserver exportresourcestrings de c:\Strings.xml
```

- El primer comando crea un archivo llamado Strings.xml en la carpeta c:\; ese archivo contiene las cadenas de los recursos de RaptorXML Server en inglés.
- El segundo comando llama al ejecutable del servidor y hace lo mismo que el primero.

Localizar RaptorXML Server en otros idiomas

Si quiere puede localizar RaptorXML Server en cualquier idioma. Altova ya ofrece la aplicación en cinco idiomas: inglés, español, francés, alemán y japonés (todos los archivos están en la carpeta C:\Program Files (x86)\Altova\RaptorXMLServer2024\bin) pero puede localizarla en cualquier otro idioma.

Siga estos pasos para localizar la aplicación:

1. Genere un archivo XML con las cadenas de recursos usando el comando exportresourcestrings (*ver más arriba*). Las cadenas de recursos de este archivo XML puede estar en uno de estos idiomas: inglés (en), español (es), francés (fr), alemán (de) o japonés (ja), dependiendo del argumento ***CódigoIdioma*** que utilice con el comando.
2. Traduzca las cadenas de recursos al idioma de destino. Las cadenas de recursos son el contenido de los elementos <string> del archivo XML. No traduzca las variables que aparecen entre llaves, p. ej. {option} o {product}.
3. Póngase en contacto con el equipo de soporte técnico de Altova, que le ayudarán a generar un archivo DLL localizado de RaptorXML Server a partir de su archivo XML traducido.
4. Cuando reciba el archivo DLL localizado del equipo de soporte técnico de Altova, guárdelo en la carpeta C:\Archivos de programas (x86)\Altova\RaptorXMLServer2024\bin. El DLL tendrá un

 © 2017-2023 Altova GmbH

nombre similar a este `RaptorXML2024_ci.dll`. La parte `_ci` del nombre contiene el código del idioma. Por ejemplo, en `RaptorXML2024_de.dll`, la parte `de` es el código del idioma alemán (Deutsch).

1. Ejecute el comando `setdeflang` para establecer el archivo DLL localizado como aplicación RaptorXML Server predeterminada. Use el código de idioma del nombre del archivo DLL como argumento del comando `setdeflang`

Nota: Altova ya ofrece RaptorXML Server en estos cinco idiomas: inglés, español, francés, alemán y japonés. Para usar uno de estos idiomas como idioma predeterminado, use el comando `setdeflang` de RaptorXML Server.

5.10.10 debug

Sintaxis y descripción

El comando `debug` inicia RaptorXML Server para procesos de depuración y no como servicio. Para detener este modo de RaptorXML Server basta con pulsar **Ctrl+C**.

```
raptorxmlserver debug [opciones]
```

▼ Mayúsculas/minúsculas y barras en la línea de comandos

`RaptorXML` (y **`RaptorXMLServer`** para comandos administrativos) *en Windows*
`raptorxml` (y **`raptorxmlserver`** para comandos administrativos) *en Windows y Unix (Linux, Mac)*

* Las minúsculas (`raptorxml` y `raptorxmlserver`) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que las mayúsculas (`RaptorXML`) funcionan solamente en Windows y Mac.
* Use la barra diagonal en Linux y Mac y la barra diagonal inversa en Windows.

▼ Barra diagonal inversa y espacios en sistemas Windows

En sistemas Windows: si hay espacios o caracteres especiales (por ejemplo en los nombres de archivos o de carpetas, empresas, personas o productos) debe usar comillas: por ejemplo, **`"Mi archivo"`**. Sin embargo, debe tener en cuenta que una barra diagonal inversa seguida de comillas dobles (por ejemplo, **`"C:\Mi Directorio\"`**) es posible que no se lea correctamente. Esto se debe a que la barra diagonal inversa también se usa para empezar secuencias de escape y la secuencia de escape **`\"`** también usa las comillas dobles. Para evitar secuencia de caracteres puede añadir otra barra diagonal inversa: **`\\"`**. En resumen: si necesita escribir una ruta que contenga espacios y una barra diagonal inversa, esta es la mejor forma de hacerlo: **`"C:\Mi Directorio\\"`**.

Ejemplo

Ejemplo del comando `debug`:

```
raptorxmlserver debug
```

Opciones

Las opciones se enumeran en versión corta (si existe) y larga. Puede usar una o dos barras tanto para la versión corta como para la larga. Las opciones pueden tomar un valor o no hacerlo. Si lo hacen, se escribe así: **`--option=value`**. Los valores se pueden indicar sin comillas, salvo en dos casos: (i) cuando la cadena del

valor contiene espacios o (ii) cuando se indica de forma explícita en la descripción de la opción que las comillas son necesarias. Si una opción toma un valor booleano y no se especifica ningún valor, entonces el valor predeterminado de la opción es `TRUE`. Use la opción **`--h, --help`** para mostrar información sobre el comando.

- config [c]

  `--c, --config = File`
  Indica la ruta a un archivo de configuración.

- port

  `--port = PortNumber`
  El número de puerto de la instancia de depuración de RaptorXML Server

5.10.11 help

Sintaxis y descripción

El comando **`help`** toma un único argumento (`Command`), que es el nombre del comando para el que necesita la ayuda, y muestra la sintaxis del comando, sus opciones y otra información relevante. Si no se especifica el comando `Command`, entonces se enumeran todos los comandos del ejecutable, cada uno con una breve descripción.Puede llamar al comando **`help`** desde estos ejecutables: **`raptorxml`** o **`raptorxmlserver`**.

```
raptorxml help Command
raptorxmlserver help Command
```

- Mayúsculas/minúsculas y barras en la línea de comandos

  `RaptorXML` (y **`RaptorXMLServer`** para comandos administrativos) *en Windows*
  `raptorxml` (y **`raptorxmlserver`** para comandos administrativos) *en Windows y Unix (Linux, Mac)*

  * Las minúsculas (`raptorxml` y `raptorxmlserver`) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que las mayúsculas (`RaptorXML`) funcionan solamente en Windows y Mac.
  * Use la barra diagonal en Linux y Mac y la barra diagonal inversa en Windows.

Ejemplo

Ejemplos del comando **`help`** para mostrar información sobre el comando **`licenseserver`** (este comando está en los dos ejecutables):

```
raptorxml help licenseserver
raptorxmlserver help licenseserver
```

La opción --help

También puede acceder a la información de ayuda sobre un comando usando la opción `--help` del comando para el que necesita la ayuda. Los comandos siguientes obtienen el mismo resultado:

 © 2017-2023 Altova GmbH

```
raptorxml licenseserver --help
```

El comando anterior usa la opción `--help` del comando **`licenseserver`**.

```
raptorxml help licenseserver
```

El comando **`help`** toma como argumento `licenseserver`.

En ambos casos, aparece información de ayuda sobre el comando **`licenseserver`**.

5.10.12 version

Sintaxis y descripción

El comando **`version`** muestra el número de versión de RaptorXML Server.Se puede llamar a este comando desde estos ejecutables: **`raptorxml`** o **`raptorxmlserver`**.

```
raptorxml version
raptorxmlserver version
```

▼ Mayúsculas/minúsculas y barras en la línea de comandos

`RaptorXML` (y **`RaptorXMLServer`** para comandos administrativos) *en Windows*
`raptorxml` (y **`raptorxmlserver`** para comandos administrativos) *en Windows y Unix (Linux, Mac)*

* Las minúsculas (`raptorxml` y `raptorxmlserver`) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que las mayúsculas (`RaptorXML`) funcionan solamente en Windows y Mac.
* Use la barra diagonal en Linux y Mac y la barra diagonal inversa en Windows.

Ejemplo

Ejemplos del comando `version`:

```
raptorxml version
raptorxmlserver version
```

5.11 Opciones

En esta sección describimos todas las opciones de la ILC, organizadas por funciones. Para ver qué opción debe usar con cada comando, consulte la descripción del comando correspondiente.

- Catálogos, recursos globales y archivos ZIP 236
- Mensajes, errores y ayuda 237
- Procesamiento 238
- XML 239
- XSD 240
- XQuery 243
- XSLT 245
- JSON/Avro 247
- XML Signatures 248

5.11.1 Catálogos, recursos globales, archivos ZIP

▼ catalog

--catalog = *ARCHIVO*
Especifica la ruta de acceso absoluta a un archivo de catálogo que no está en el archivo de catálogo raíz instalado. El valor predeterminado es la ruta de acceso absoluta del archivo de catálogo raíz instalado (`<carpeta-instalación>\Altova\RaptorXMLServer2024\etc\RootCatalog.xml`). Consulte el apartado Catálogos XML 48 para obtener más información.

▼ user-catalog

--user-catalog = *ARCHIVO*
Especifica la ruta de acceso absoluta a un catálogo XML que debe utilizarse junto con el catálogo raíz. Consulte el apartado Catálogos XML 48 para obtener más información.

▼ enable-globalresources

--enable-globalresources = `true|false`
Habilita la función de recursos globales 55. Valor predeterminado: `false`.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

▼ globalresourceconfig [gc]

--gc | --globalresourceconfig = *VALOR*
Especifica la configuración activa del recurso global 55 (y habilita los recursos globales) 55.

▼ globalresourcefile [gr]

--gr | --globalresourcefile = *ARCHIVO*
Especifica el archivo de recursos globales 55 (y habilita los recursos globales) 55.

▼ recurse

--recurse = `true|false`

© 2017-2023 Altova GmbH

Esta opción se utiliza para seleccionar ficheros dentro de subdirectorios (incluso en archivos ZIP). Si el valor es `true`, el argumento *`ArchivoEntrada`* del comando seleccionará el fichero seleccionado también en los subdirectorios. Por ejemplo: `"test.zip|zip\test.xml"` seleccionará los ficheros llamados `test.xml` en todos los subdirectorios de la carpeta ZIP. Si quiere puede usar los caracteres comodín `*` y `?`. Por ejemplo: `*.xml` seleccionaría todos los ficheros de la carpeta ZIP que tengan la extensión `.xml`. Valor predeterminado: `false`
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

5.11.2 Mensajes, errores, ayuda, tiempo de espera y versión

- error-format

  `--error-format =` `text|shortxml|longxml`
  Especifica el formato de la salida de error. Los valores posibles son formatos de texto, XML y XML detallado (`longxml`). Valor predeterminado: `text`.

- error-limit

  `--error-limit =` *`N | unlimited`*
  Especifica el límite de errores con un valor comprendido entre `1` y `9999` o `unlimited` (ilimitado). El valor predeterminado es `100`. Cuando se alcanza el límite de error, se detiene la validación. Esta opción es muy práctica a la hora de limitar el uso del procesador durante la validación o transformación.

- help

  `--help`
  Muestra el texto de ayuda para el comando. Por ejemplo `valany --h`. (Otra opción es usar el comando `help` con un argumento. Por ejemplo: `help valany`).

- info-limit

  `--info-limit =` *`N | unlimited`*
  Indica el límite del mensaje de información dentro del rango `1-65535` or `unlimited`. Si se alcanza el límite de información indicado, el procesamiento continúa pero ya no se informa de más mensajes. El valor predeterminado es `100`.

- log-output

  `--log-output =` *`ARCHIVO`*
  Escribe el registro de salida en la URL de archivo indicada. Compruebe que la ILC tiene permiso de escritura en la ubicación de destino.

- network-timeout

  `--network-timeout =` *`VALOR`*
  Indica el tiempo de espera en milisegundos para operaciones remotas de entrada y salida. Valor predeterminado: `40000`.

- verbose

  `--verbose =` `true|false`

Si el valor es `true`, se genera información adicional durante la validación. Valor predeterminado es `false`.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- verbose-output

  `--verbose-output =` *`ARCHIVO`*
  Escribe el resultado detallado en el *`ARCHIVO`* indicado.

- version

  `--version`
  Muestra el número de versión de RaptorXML Server. Si se utiliza con un comando, escriba la opción `--version` antes del comando.

- warning-limit

  `--warning-limit =` *`N | unlimited`*
  Especifica el límite de advertencia en el rango `1-65535` o `unlimited` (ilimitado). El procesamiento continua si se alcanza el límite pero no se registrarán más advertencias. El valor por defecto es `100`.

5.11.3 Procesamiento

- listfile

  `--listfile =` `true|false`
  Si el valor es `true`, el argumento *`ArchivoEntrada`* del comando se entiende como un archivo de texto que contiene un nombre de archivo por línea. Otra opción es enumerar los archivos en la ILC, separados por un espacio. No obstante, recuerde que las ILC tienen un límite de caracteres. Además, no olvide que la opción `--listfile` solamente afecta a los argumentos y no a las opciones.
  Valor predeterminado: `false`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- parallel-assessment [pa]

  `--pa | --parallel-assessment =` `true|false`
  Si el valor es `true`, la evaluación de la validez de esquemas se realiza en paralelo. Esto significa que si hay más de 128 elementos en cualquiera de los niveles, estos elementos se procesan en paralelo utilizando varios subprocesos. Por tanto, los archivos XML de gran tamaño se pueden procesar más rápido si se habilita esta opción. La evaluación en paralelo se lleva cabo nivel por nivel, pero puede ocurrir en varios niveles de un mismo conjunto de información. Recuerde que la evaluación en paralelo no funciona en modo de transmisión por secuencias. Por este motivo la opción `--streaming` se pasa por alto si el valor de la opción `--parallel-assessment` es `true`. Además, se usa más memoria cuando se utiliza la opción `--parallel-assessment`. El valor predeterminado de esta opción es false y su forma abreviada es `--pa`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- script

  `--script =` *`ARCHIVO`*
  Una vez finalizada la validación, ejecuta el script Python. Para indicar más de un script basta con agregar

 © 2017-2023 Altova GmbH

la opción varias veces.

- script-api-version

  --api, --script-api-version = `1; 2; 2.1 a 2.4; 2.4.1; 2.5 to 2.8; 2.8.1 a 2.8.6`
  Especifica la versión de la API de Python a utilizar para el script. El valor predeterminado es la última versión, actualmente es **2.8.6**. En lugar de valores numéricos enteros como `1` y `2`, también puede utilizar los valores correspondientes `1.0` y `2.0`. Asimismo, puede utilizar el número de tres dígitos `2.5.0` en vez de dos (`2.5`). Consulte también el apartado Versiones de la API de Python 372.

- script-param

  --script-param = *CLAVE:VALOR*
  Parámetros definidos por el usuario a los que se puede acceder durante la ejecución de scripts Python.

- streaming

  --streaming = `true|false`
  Habilita la transmisión por secuencias. En el modo de transmisión por secuencias, el almacenamiento de datos en memoria se reduce al mínimo y el procesamiento es más rápido. El inconveniente es que puede que no esté disponible cierta información que podría necesitar más adelante, como el modelo de datos del documento XML, por ejemplo. Si quiere evitar esto, debería deshabilitar el modo de transmisión por secuencias (dándole el valor `false` a la opción `--streaming`). Cuando use la opción `--script` con el comando `valxml-withxsd`, aconsejamos deshabilitar la transmisión por secuencias. Recuerde que la opción `--streaming` se ignora, si el valor de `--parallel-assessment` es `true`.
  Valor predeterminado: `true`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- xml-validation-error-as-warning

  --xml-validation-error-as-warning = `true|false`
  Si es `true`, tratar los errores de validación como advertencias. Si los errores se tratan como advertencias, el procesamiento adicional como la transformación XSLT, proseguirá independientemente de los errores. Por defecto es `false`.

5.11.4 XML

- assessment-mode

  --assessment-mode = `lax|strict`
  Especifica el modo de evaluación de la validez del esquema, según se define en las especificaciones XSD. El documento XML de instancia se validará en función del modo especificado en esta opción. Valor predeterminado: `strict`.

- dtd

  --dtd = *ARCHIVO*
  Especifica el documento DTD externo que debe utilizarse para la validación. Si en el documento XML hay una referencia a una DTD externa, esta opción de la ILC reemplaza a la referencia externa.

- load-xml-with-psvi

  --load-xml-with-psvi = `true|false`
  Habilita la validación de archivos XML de entrada y genera información posterior a la validación. Valor predeterminado: `true`.

- namespaces

  --namespaces = `true|false`
  Habilita el procesamiento preparado para espacios de nombres. Esta opción es muy útil si quiere buscar en la instancia XML errores resultantes de espacios de nombres erróneos.
  Valor predeterminado: `false`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- xinclude

  --xinclude = `true|false`
  Habilita la compatibilidad con inclusiones XML (XInclude). Si el valor es `false`, los elementos XInclude `include` se ignoran. Valor predeterminado: `false`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- xml-mode

  --xml-mode = `wf|id|valid`
  Especifica el modo de procesamiento XML que debe utilizarse para el documento de instancia XML: `wf`=comprobación de formato; `id`=comprobación de formato con ID/IDREF; `valid`=validación. Valor predeterminado: `wf`. Recuerde que el valor `valid` exige que cada documento de instancia que se cargue durante el procesamiento haga referencia a una DTD. Si no existe ninguna DTD, se generará un error.

- xml-validation-error-as-warning

  --xml-validation-error-as-warning = `true|false`
  Si es `true`, tratar los errores de validación como advertencias. Si los errores se tratan como advertencias, el procesamiento adicional como la transformación XSLT, proseguirá independientemente de los errores. Por defecto es `false`.

- xsd

  --xsd = *ARCHIVO*
  Especifica qué esquemas XML deben utilizarse para la validación de documentos XML. Si quiere especificar más de un esquema, añada la opción varias veces.

5.11.5 XSD

- assessment-mode

  --assessment-mode = `lax|strict`
  Especifica el modo de evaluación de la validez del esquema, según se define en las especificaciones XSD. El documento XML de instancia se validará en función del modo especificado en esta opción. Valor predeterminado: `strict`.

 © 2017-2023 Altova GmbH

▼ ct-restrict-mode

--ct-restrict-mode = 1.0|1.1|default
Especifica cómo comprobar restricciones de tipo complejo. Un valor de **1.0** comprueba restricciones de tipo complejo conforme a lo definido en la especificación XSD 1.0 (incluso estando en modo de validación XSD 1.1). Un valor de **1.1** comprueba restricciones de tipo complejo conforme a los definido en la especificación XSD 1.1 (incluso estando en modo de validación XSD 1.0). Un valor de **default** comprueba restricciones de tipo complejo conforme a lo definido en la especificación XSD del modo de validación habilitado en ese momento (1.0 o 1.1). El valor predeterminado es **default**.

▼ namespaces

--namespaces = true|false
Habilita el procesamiento preparado para espacios de nombres. Esta opción es muy útil si quiere buscar en la instancia XML errores resultantes de espacios de nombres erróneos.
Valor predeterminado: false.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en true.

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

--report-import-namespace-mismatch-as-warning = true|false
Informa de los errores de disparidad del espacio de nombres o del espacio de nombres de destino al importar esquemas con **xs:import** como advertencias en vez de como errores. El valor predeterminado es false.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en true.

▼ schema-imports

--schema-imports = load-by-schemalocation | load-preferring-schemalocation | load-by-namespace | load-combining-both | license-namespace-only
Esta opción indica el comportamiento de los elementos xs:import. Cada uno de estos elementos tiene un atributo opcional namespace y un atributo opcional schemaLocation: <import namespace="unEspacioNombres" schemaLocation="unaURL">. La opción indica si se debe cargar un documento de esquema o solo autorizar a un espacio de nombres. Si la opción indica que se debe cargar un documento de esquema, entonces indica también qué información debe utilizarse para encontrar el documento de esquema. Valor predeterminado: load-preferring-schemalocation.

- load-by-schemalocation: el valor del atributo schemaLocation se utiliza para buscar el esquema, teniendo en cuenta las asignaciones de catálogo[48]. Si está presente el atributo namespace, se importa el espacio de nombres (con licencia).
- load-preferring-schemalocation: si está presente, se utiliza el atributo schemaLocation teniendo en cuenta las asignaciones de catálogo[48]. Si no está presente el atributo schemaLocation, entonces se usa el valor del atributo namespace a través de las asignaciones de catálogo[48]. Este es el **valor predeterminado**.
- load-by-namespace: el valor del atributo namespace se utiliza para buscar el esquema por medio de una asignación de catálogo[48].
- load-combining-both: si el atributo namespace o schemaLocation tiene una asignación de catálogo[48], entonces se usa la asignación. Si ambos atributos tienen asignaciones de catálogo[48], entonces es el valor de la opción --schema-mapping (opción XML/XSD[240]) decide qué asignación se utiliza. Si no hay ninguna asignación de catálogo[48], entonces se usa el atributo schemaLocation.
- license-namespace-only: se importa el espacio de nombres. No se importa el documento de esquema.

▼ schemalocation-hints

--schemalocation-hints = `load-by-schemalocation | load-by-namespace | load-combining-both | ignore`

Determina el comportamiento predeterminado de los atributos `xsi:schemaLocation` y `xsi:noNamespaceSchemaLocation`. Indica si se debe cargar un documento de esquema y, si así es, indica qué información debe utilizarse para encontrarlo. Valor predeterminado: `load-by-schemalocation`.

- **Valor predeterminado**: `load-by-schemalocation`. Este valor toma la URL de la ubicación del esquema[403] de los atributos `xsi:schemaLocation` y `xsi:noNamespaceSchemaLocation` de los documentos de instancia XML.
- El valor `load-by-namespace` toma la parte de espacio de nombres[403] del atributo `xsi:schemaLocation` y una cadena vacía en el caso del atributo `xsi:noNamespaceSchemaLocation` y encuentra el esquema por medio de una asignación de catálogo[48].
- Si usa el valor `load-combining-both` y el espacio de nombres o la URL tienen una asignación de catálogo[48], se usa dicha asignación. Si ambos tienen asignaciones de catálogo[48], el valor de la opción `schema-mapping` (opción XML/XSD[240]) decide qué asignación se utiliza. Si ni el espacio de nombres ni la URL tiene una asignación de catálogo, se usa la URL.
- El valor `ignore` ignora los atributos `xsi:schemaLocation` y `xsi:noNamespaceSchemaLocation`.

▼ schema-mapping

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

Si se usa la ubicación y el espacio de nombres para buscar el documento de esquema, esta opción indica cuál de ellos debe ser la opción preferida durante la búsqueda en el catálogo.
Si la opción `--schemalocation-hints` o la opción `--schema-imports` tiene el valor `load-combining-both` y si las partes de espacio de nombres y URL pertinentes tienen asignaciones de catálogo[48], entonces el valor de la opción especifica cuál de las dos asignaciones se utiliza (la asignación del espacio de nombres o de la URL: el valor `prefer-schemalocation` se refiere a la asignación de la URL). Valor predeterminado: `prefer-schemalocation`.

▼ xml-mode-for-schemas

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

Especifica el modo de procesamiento XML que debe utilizarse para el documento de instancia XML: `wf`=comprobación de formato; `id`=comprobación de formato con ID/IDREF; `valid`=validación. Valor predeterminado: `wf`. Recuerde que el valor `valid` exige que cada documento de esquema que se cargue durante el procesamiento haga referencia a una DTD. Si no existe ninguna DTD, se generará un error.

▼ xsd-version

--xsd-version = `1.0|1.1|detect`

Especifica qué versión de la especificación Schema Definition Language (XSD) del W3C se debe usar. Valor predeterminado: `1.0`.

Esta opción también puede ser útil si quiere ver en qué aspectos no es compatible un esquema 1.0 con la especificación 1.1. El valor `detect` es una característica de Altova. Permite detectar la versión del esquema XML (`1.0` o `1.1`) leyendo el valor del atributo `vc:minVersion` del elemento `<xs:schema>` del documento. Si el valor del atributo `@vc:minVersion` es `1.1`, se entiende que la versión del esquema es `1.1`. Si el atributo tiene otro valor que no sea `1.1` (o si no está presente el atributo `@vc:minVersion`), se entiende que la versión del esquema es `1.0`.

 © 2017-2023 Altova GmbH

5.11.6 XQuery

- indent-characters

  --indent-characters = *VALOR*
  Especifica la cadena de caracteres que debe usarse como sangría.

- input

  --input = *ARCHIVO*
  La URL del archivo XML que se debe transformar.

- keep-formatting

  --keep-formatting = `true|false`
  Conserva en la medida de lo posible el formato del documento de destino. Valor predeterminado: `true`.

- omit-xml-declaration

  --omit-xml-declaration = `true|false`
  Opción de serialización que especifica si la declaración XML se omite en el resultado o no. Si el valor es `true`, el documento de salida no tendrá una declaración XML. Si el valor es `false`, se incluye una declaración XML en el documento de salida. Valor predeterminado: `false`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- output, xsltoutput

  output = *ARCHIVO,* **xsltoutput =** *ARCHIVO*
  La URL del archivo de salida principal. Por ejemplo, en caso de tener varios archivos HTML de salida, el archivo de salida principal será la ubicación del archivo HTML del punto de entrada. Los demás archivos de salida (como archivos de imagen generados) se indican con `xslt-additional-output-files`. Si no se especifica la opción `--output` ni la opción `--xsloutput`, se genera un resultado estándar.

- output-encoding

  --output-encoding = *VALOR*
  El valor del atributo `encoding` del documento de salida. Son valores válidos todos los nombres del registro de juego de caracteres IANA. Valor predeterminado: `UTF-8`.

- output-indent

  --output-indent = `true|false`
  Si el valor es `true`, la sangría del documento de salida seguirá su estructura jerárquica. Si el valor es `false`, el documento de salida no tendrá sangría jerárquica. Valor predeterminado: `false`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- output-method

  --output-method = `xml|html|xhtml|text`
  Especifica el formato de salida. Valor predeterminado: `xml`.

- param [p]

  --p | --param = *CLAVE:VALOR*

XQuery

Especifica el valor de un parámetro externo. En el documento XQuery los parámetros externos se declaran con la declaración `declare variable` seguida de un nombre de variable y después la palabra clave `external` seguida del punto y coma final. Por ejemplo: `declare variable $foo as xs:string external;`

Al usar la palabra clave `external`, `$foo` se convierte en parámetro externo y su valor se pasa en tiempo de ejecución desde una fuente externa. El parámetro externo recibe un valor con el comando de la ILC. Por ejemplo: `--param=foo:'MiNombre'`

En la descripción anterior, *CLAVE* es el nombre de parámetro externo y *VALOR* es su valor, dado como expresión XPath. Los nombres de parámetro utilizados en la ILC deben declararse en el documento XQuery. Si se pasan valores a varios parámetros externos en la ILC, cada parámetro debe llevar una opción `--param` distinta. Si la expresión XPath contiene espacios, entonces debe estar entre comillas dobles.

XSLT

Especifica un parámetro global de la hoja de estilos. *CLAVE* es el nombre del parámetro y *VALOR* es una expresión XPath que da un valor al parámetro. Los nombres de parámetro utilizados en la ILC deben declararse en la hoja de estilos. Si usa más de un parámetro, debe usar el modificador `--param` antes de cada parámetro. Si la expresión XPath incluye espacios, entonces debe ir entre comillas dobles, tanto si el espacio está en la expresión propiamente dicha o en un literal de cadena de la expresión. Por ejemplo:

```
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`

Indica qué se debe hacer con el archivo XML actualizado.

- discard: las actualizaciones se descartan y no se escriben en el archivo. Ni el archivo de entrada ni el de salida se actualizan. Este es el valor predeterminado.
- writeback: escribe las actualizaciones en el archivo XML de entrada indicado en la opción `--input`.
- asmainresult: escribe las actualizaciones en el archivo XML de salida indicado en la opción `--output`. Si no se indicó la opción `--output`, las actualizaciones se escriben en el archivo estándar de salida. En ambos casos el archivo XML no se modifica.

Valor predeterminado: `discard`.

xpath-static-type-errors-as-warnings

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

Si se establece en `true` reduce a advertencia cualquier error que se detecte en el contexto estático XPath. Mientras que un error provocaría un error, una advertencia permite que el procesamiento continúe. El valor predeterminado es `false`.

xquery-update-version

 © 2017-2023 Altova GmbH

--xquery-update-version = 1|1.0|3|3.0|
Indica si el procesador XQuery debería usar XQuery Update Facility 1.0 o XQuery Update Facility 3.0. Valor predeterminado: 3.

▼ xquery-version

--xquery-version = 1|1.0|3|3.0|3.1
Indica si el procesador XQuery debe usar XQuery 1.0 o 3.0. Valor predeterminado: 3.1

5.11.7 XSLT

▼ chartext-disable

--chartext-disable = true|false
Deshabilita las extensiones de gráficos. Valor predeterminado: false.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en true.

▼ dotnetext-disable

--dotnetext-disable = true|false
Deshabilita las extensiones .NET. Valor predeterminado: false.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en true.

▼ indent-characters

--indent-characters = *VALOR*
Especifica la cadena de caracteres que debe usarse como sangría.

▼ input

--input = *ARCHIVO*
La URL del archivo XML que se debe transformar.

▼ javaext-barcode-location

--javaext-barcode-location = *ARCHIVO*
Especifica la ruta de acceso de la carpeta que contiene el archivo de extensión de código de barras AltovaBarcodeExtension.jar. La ruta de acceso debe darse en uno de estos formatos:

- Un URI de archivo (ejemplo: --javaext-barcode-location="file:///C:/Archivos de programa/Altova/RaptorXMLServer2024/etc/jar/")
- Una ruta de acceso Windows con caracteres de escape para las barras diagonales inversas (ejemplo: --javaext-barcode-location="C:\\Archivos de programa\\Altova\\RaptorXMLServer2024\\etc\\jar\\")

▼ javaext-disable

--javaext-disable = true|false
Deshabilita las extensiones Java. Valor predeterminado: false.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en true.

▼ output, xsltoutput

output = *ARCHIVO*, **xsltoutput =** *ARCHIVO*

La URL del archivo de salida principal. Por ejemplo, en caso de tener varios archivos HTML de salida, el archivo de salida principal será la ubicación del archivo HTML del punto de entrada. Los demás archivos de salida (como archivos de imagen generados) se indican con `xslt-additional-output-files`. Si no se especifica la opción `--output` ni la opción `--xsloutput`, se genera un resultado estándar.

▼ param [p]

--p | --param = *CLAVE:VALOR*

⊟ ***XQuery***

Especifica el valor de un parámetro externo. En el documento XQuery los parámetros externos se declaran con la declaración `declare variable` seguida de un nombre de variable y después la palabra clave `external` seguida del punto y coma final. Por ejemplo: `declare variable $foo as xs:string external;`

Al usar la palabra clave `external`, `$foo` se convierte en parámetro externo y su valor se pasa en tiempo de ejecución desde una fuente externa. El parámetro externo recibe un valor con el comando de la ILC. Por ejemplo: `--param=foo:'MiNombre'`

En la descripción anterior, *CLAVE* es el nombre de parámetro externo y *VALOR* es su valor, dado como expresión XPath. Los nombres de parámetro utilizados en la ILC deben declararse en el documento XQuery. Si se pasan valores a varios parámetros externos en la ILC, cada parámetro debe llevar una opción `--param` distinta. Si la expresión XPath contiene espacios, entonces debe estar entre comillas dobles.

⊟ ***XSLT***

Especifica un parámetro global de la hoja de estilos. *CLAVE* es el nombre del parámetro y *VALOR* es una expresión XPath que da un valor al parámetro. Los nombres de parámetro utilizados en la ILC deben declararse en la hoja de estilos. Si usa más de un parámetro, debe usar el modificador `--param` antes de cada parámetro. Si la expresión XPath incluye espacios, entonces debe ir entre comillas dobles, tanto si el espacio está en la expresión propiamente dicha o en un literal de cadena de la expresión. Por ejemplo:

```
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`

Habilita la transmisión por secuencias. En el modo de transmisión por secuencias, el almacenamiento de datos en memoria se reduce al mínimo y el procesamiento es más rápido. El inconveniente es que puede que no esté disponible cierta información que podría necesitar más adelante, como el modelo de datos del documento XML, por ejemplo. Si quiere evitar esto, debería deshabilitar el modo de transmisión por secuencias (dándole el valor `false` a la opción `--streaming`). Cuando use la opción `--script` con el comando `valxml-withxsd`, aconsejamos deshabilitar la transmisión por secuencias. Recuerde que la opción `--streaming` se ignora, si el valor de `--parallel-assessment` es `true`.
Valor predeterminado: `true`.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

 © 2017-2023 Altova GmbH

- initial-template, template-entry-point

  --initial-template, --template-entry-point = *VALOR*
  Indica el nombre de una plantilla con nombre de la hoja de estilos XSLT que sirve de punto de entrada de la transformación.

- initial-mode, template-mode

  --initial-mode, --template-mode = *VALOR*
  Especifica el modo de plantilla que debe usarse para la transformación.

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

  --xpath-static-type-errors-as-warnings = `true|falsetrue|false`
  Si se establece en `true` reduce a advertencia cualquier error que se detecte en el contexto estático XPath. Mientras que un error provocaría un error, una advertencia permite que el procesamiento continúe. El valor predeterminado es `false`.

- xslt-version

  --xslt-version = `1|1.0|2|2.0|3|3.0|3.1`
  Especifica si el procesador XSLT debe usar XSLT 1.0, XSLT 2.0 o XSLT 3.0. Valor predeterminado: `3`

5.11.8 JSON/Avro

- additional-schema

  --additional-schema = *FILE*
  Indica los URIs de un documento de esquema suplementario. El esquema suplementario se carga desde el principal y se puede hacer referencia a él también desde el esquema principal con la propiedad **id** o **$id** del esquema suplementario.

- disable-format-checks

  --disable-format-checks = `true|false`
  Deshabilita la validación semántica impuesta por el atributo de formato. El valor predeterminado es `false`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- jsonc

  --jsonc = `true|false`
  Habilita la compatibilidad con los comentarios en JSON. El valor predeterminado es `false`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- json-lines

  --json-lines = `true|false`
  Habilita la compatibilidad con Líneas JSON (es decir, un valor JSON por línea). El valor predeterminado es `false`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

5.11.9 Firmas XML

- absolute-reference-uri

  --absolute-reference-uri = `true|false`
  Especifica si el URI del documento firmado debe leerse como absoluto (`true`) o relativo (`false`). El valor predeterminado es `false`.
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

- certname, certificate-name

  --certname, --certificate-name = *VALOR*
  El nombre del certificado utilizado para firmar.

  Windows
  Se trata del nombre `Subject` de un certificado del almacén de certificados (**--certificate-store**) seleccionado.

  iemplo para enumerar los certificados (bajo PowerShell)

```
% ls cert://UsuarioActual/My
PSParentPath: Microsoft.PowerShell.Security\Certificate::UsuarioActual\My
Thumbprint Subject
---------- -------
C9DF64BB0AAF5FA73474D78B7CCFFC37C95BFC6C CN=certificate1
... CN=...
```

  iemplo: **--certificate-name==certificado1**

  Linux/macOS
  --certname especifica el nombre de archivo de un certificado X.509v3 con cifrado PEM con la clave privada. Estos archivos suelen tener la extensión **.pem**.

  iemplo: **--certificate-name==/ruta/de/certificado1.pem**

- certstore, certificate-store

  --certstore, --certificate-store = *VALOR*
  La ubicación donde está almacenado el certificado especificado con **--certificate-name**.

  Windows
  Nombre de un almacén de certificados situado dentro de **cert://UsuarioActual**. Los almacenes de certificados disponibles se pueden enumerar (bajo PowerShell) usando **% ls cert://UsuarioActual/**.
  Los certificados se enumerarán de la siguiente manera:

```
Nombre : TrustedPublisher
Nombre : ClientAuthIssuer
Nombre : Root
Nombre : UserDS
Nombre : CA
Nombre : ACRS
```

 © 2017-2023 Altova GmbH

```
Nombre : REQUEST
Nombre : AuthRoot
Nombre : MSIEHistoryJournal
Nombre : TrustedPeople
Nombre : MyCertStore
Nombre : Local NonRemovable Certificates
Nombre : SmartCardRoot
Nombre : Trust
Nombre : Disallowed
```

iemplo: **--certificate-store==MiAlmacénCertificados**

Linux/MacOS
La opción **--certstore** no es compatible por ahora.

▼ digest, digest-method

--digest, --digest-method = sha1|sha256|sha384|sha512
El algoritmo que se usa para computar el valor del resumen (digest) dentro del archivo XML de entrada. Los valores disponibles son: sha1|sha256|sha384|sha512.

▼ hmackey, hmac-secret-key

--hmackey, --hmac-secret-key = *VALOR*
La clave secreta compartida HMAC. Debe tener una longitud mínima de seis caracteres.

Ejemplo: **--hmackey=contraseñaSecreta**

▼ hmaclen, hmac-output-length

--hmaclen, --hmac-output-length = *LONGITUD*

Trunca el resultado del algoritmo HMAC hasta el número de bits indicado por LONGITUD. Si se especifica, este valor debe ser:
- múltiplo de 8,
- mayor que 80 y
- mayor que la mitad de la longitud de salida del algoritmo hash subyacente.

▼ keyinfo, append-keyinfo

--keyinfo, --append-keyinfo = true|false
Especifica si se debe incluir el elemento **KeyInfo** en la firma o no. El valor predeterminado es **false**.

▼ sigc14nmeth, signature-canonicalization-method

--sigc14nmeth, --signature-canonicalization-method = *VALOR*
Especifica el algoritmo de canonización que se debe aplicar al elemento **SignedInfo**. El valor debe ser uno de estos tres valores:

- REC-xml-c14n-20010315
- xml-c14n11
- xml-exc-c14n#

▼ sigmeth, signature-method

--sigmeth, --signature-method = *VALOR*
Especifica el algoritmo que se debe usar para generar la firma.

Cuando se usa un certificado
Si se especificó un certificado, entonces `--signature-method` es opcional y el valor para este parámetro se deriva del certificado. Debe coincidir con el algoritmo utilizado por el certificado.

iemplo: **--signature-method=rsa-sha256**

Cuando se usa --hmac-secret-key
Si se usa la opción `--hmac-secret-key`, entonces esta opción es obligatoria. El valor debe ser uno de los algoritmos HMAC compatibles:

- `hmac-sha256`
- `hmac-sha386`
- `hmac-sha512`
- `hmac-sha1` (no recomendable según la especificación)

iemplo: **--signature-method=hmac-sha256**

▼ sigtype, signature-type

--sigtype, --signature-type = `detached | enveloping | enveloped`
Especifica el tipo de firma que se debe generar (separada, envolvente o envuelta).

▼ transforms

--transforms = *VALOR*
Especifica las transformaciones XML Signature que se deben aplicar al documento de salida. Esta opción se puede especificar más de una vez. Si se especifica varias veces, el orden en que se especifica es importante. La primera transformación especificada recibe el documento de entrada. La última se utiliza inmediatamente antes de calcular el valor implícito.

Valores compatibles:

- `REC-xml-c14n-20010315` para Canonical XML 1.0 (omitir comentarios)
- `xml-c14n11` para Canonical XML 1.1 (omitir comentarios)
- `xml-exc-c14n#` para Exclusive XML Canonicalization 1.0 (omitir comentarios)
- `REC-xml-c14n-20010315#WithComments` para Canonical XML 1.0 (con comentarios)
- `xml-c14n11#WithComments` para Canonical XML 1.1 (con comentarios)
- `xml-exc-c14n#WithComments` para Exclusive XML Canonicalization 1.0 (con comentarios)
- `base64`
- extensión de Altova `strip-whitespaces`

iemplo: **--transforms=xml-c14n11**

▼ write-default-attributes

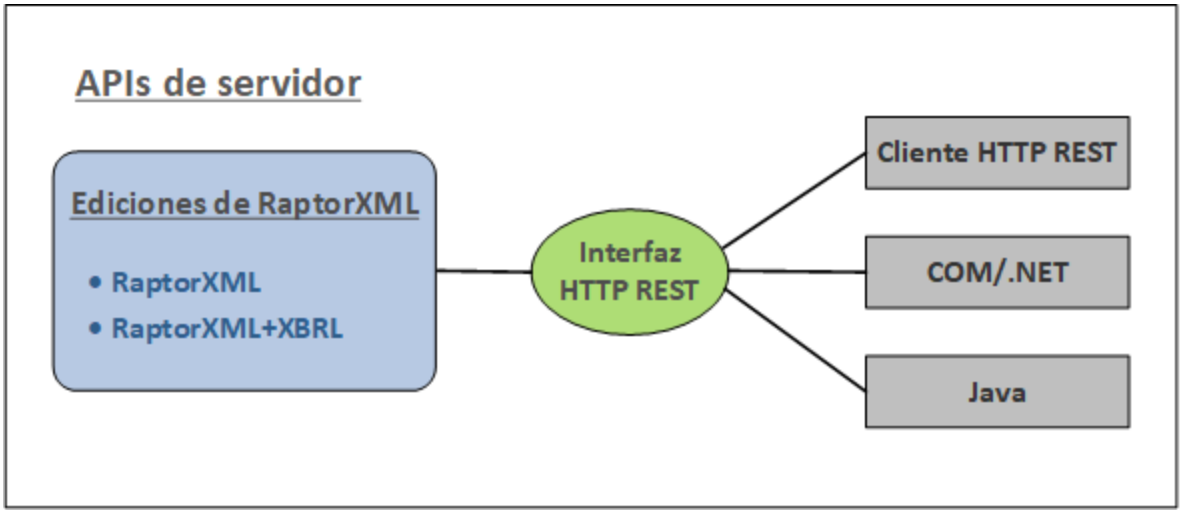
--write-default-attributes = `true|false`

 © 2017-2023 Altova GmbH

Especifica si se deben incluir los valores de atributo predeterminados de la DTD en el documento firmado.

6 APIs de servidor: HTTP REST, COM/.NET, Java

RaptorXML Server define una interfaz HTTP REST que los clientes usan para enviar trabajos al servidor. Los clientes pueden acceder a la interfaz HTTP REST directamente o usar las APIs de servidor de alto nivel de COM/.NET y Java. Estas APIs permiten usar fácilmente clases COM/.NET y Java para gestionar la creación y el envío de solicitudes HTTP REST. La imagen siguiente muestra un resumen de los métodos de HTTP REST disponibles para comunicarse con el servidor RaptorXML.



Existen tres APIs de servidor que se pueden usar para comunicarse con RaptorXML a través de la interfaz HTTP REST (*imagen anterior*).

- nterfaz cliente HTTP REST 254
- API de COM/.NET 291
- API de Java 300

Nota: las APIs de servidor ofrecen funciones similares a los de la interfaz de la linea de comandos (ILC) 58. Esto incluye la validación y la transformación de documentos. Si quiere usar las funciones avanzadas, como las de lectura, extracción y análisis de datos, entonces use las APIs de motor. Las APIs de motor pueden proporcionar información adicional, como el conteo de elementos, su posición en el documento, o el acceso a datos XBRL complejos y su manipulación.

Uso

RaptorXML Server debe estar instalado en un equipo que sea accesible por los clientes de toda la red local. Una vez se haya iniciado el servicio de RaptorXML Server, los clientes se pueden conectar al servidor y enviar solicitudes. Los siguientes métodos de acceso están etiquetados como APIs de servidor porque sirven para comunicarse con un servidor RaptorXML remoto.

- *Interfaz cliente HTTP REST* 254: las solicitudes de clientes se realizan en formato JSON y se describen en el apartado Interfaz cliente HTTP REST 254. Cada solicitud está asignada a un directorio de trabajo en el servidor en el que se guardan los archivos de salida. El servidor responde al cliente con toda la información relevante para el trabajo.
- *API de COM/.NET* 291 *y API de Java* 300: as aplicaciones y los scripts en lenguajes de programación COM/.NET 291 y las aplicaciones Java 300 usan objetos de la API de servidor de RaptorXML 303 para

 © 2017-2023 Altova GmbH

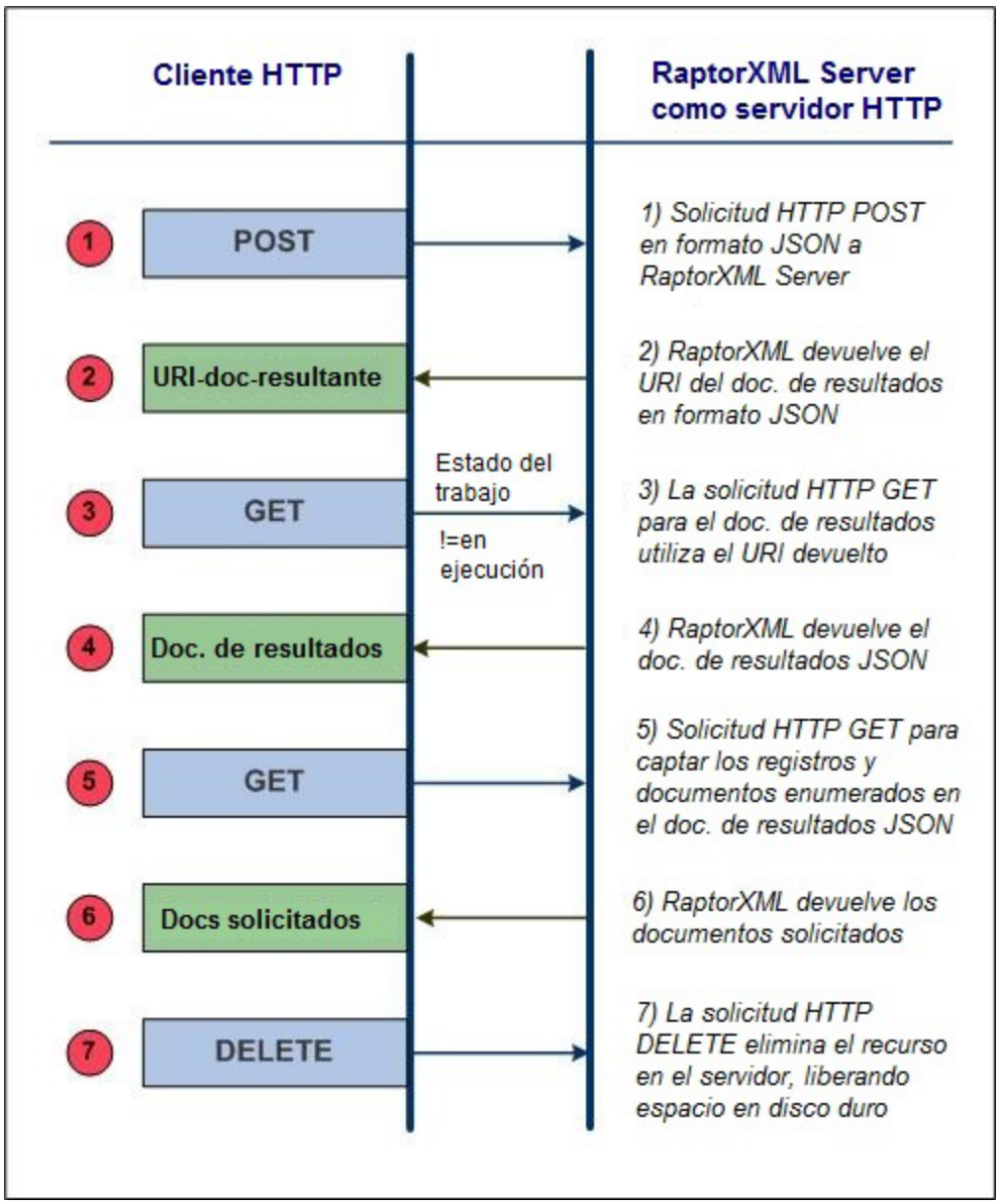
acceder a las funciones de RaptorXML Server. La API de servidor de RaptorXML 303 envía la solicitud HTTP REST correspondiente en nombre del cliente. Consulte los subapartados correspondientes para más información.

Licencias

RaptorXML Server tiene asignada una licencia en el equipo en el que está instalado. Las conexiones a RaptorXML Server funcionan vía HTTP.

6.1 Interfaz cliente HTTP REST

RaptorXML Server acepta trabajos de validación suministrados por HTTP (o HTTPS[263]). La descripción del trabajo así como los resultados se intercambian en formato JSON. El diagrama que aparece a continuación muestra el flujo de trabajo.



 © 2017-2023 Altova GmbH

Problemas de seguridad relacionados con la interfaz HTTP REST

La interfaz HTTP REST permite por defecto escribir documentos de resultados en cualquier ubicación indicada por el cliente (y a la que se pueda acceder con el protocolo HTTP). Por tanto, es importante tener en cuenta este aspecto de seguridad cuando instale y configure RaptorXML Server.

Si le preocupa que esto pueda comprometer la seguridad de su sistema o que la interfaz se utilice de forma incorrecta, puede configurar el servidor para que escriba los documentos de resultados en un directorio de salida específico del servidor mismo. Esto se consigue estableciendo el valor `false` para la opción `server.unrestricted-filesystem-access` 259 del archivo de configuración del servidor. Si se limita así el acceso, el cliente puede descargar los documentos de resultados del directorio de salida específico mediante solicitudes `GET`. Otra opción es que el administrador copie/cargue los documentos de resultados del servidor en la ubicación de destino.

Secciones de este tema

Antes de enviar una solicitud cliente, RaptorXML Server debe iniciarse y configurarse correctamente. Esto se explica en la sección Preparar el servidor 255, mientras que la sección Solicitudes cliente 267 describe cómo se envían solicitudes clientes al servidor. La sección Ejemplo en C# para API REST 287 describe el archivo de ejemplo API REST que se instala con el paquete RaptorXML Server.

6.1.1 Preparar el servidor

RaptorXML debe tener una licencia asignada en el equipo en el que está instalado. Se puede acceder a RaptorXML a través de la Interfaz HTTP REST 254. Siga las instrucciones que aparecen a continuación para preparar correctamente RaptorXML Server. Recuerde que antes debe instalarlo 47 y asignarle licencias 47.

1. Para poder acceder a RaptorXML Server por HTTP o HTTPS antes debe iniciarlo como servicio o como aplicación 256. Hay varias formas de hacerlo, dependiendo del sistema operativo: en Windows 256, en Linux 256, en macOS 257.
2. Utilice la configuración inicial del servidor 258 para probar la conexión con el servidor 257. (La configuración inicial del servidor 258 es la configuración predeterminada que viene con la instalación). Puede usar una simple solicitud HTTP `GET` como `http://localhost:8087/v1/version` para probar la conexión. (La solicitud también se puede escribir directamente en la barra de dirección del explorador). Si el servicio está en ejecución, debería recibir una respuesta a la solicitud HTTP de prueba.
3. Revise el archivo de configuración del servidor 258, `server_config.xml`. Si quiere cambiar alguna opción de configuración 259, edite el archivo y guarde los cambios. La opción HTTPS está deshabilitada por defecto y debe habilitarse en el archivo de configuración 258.
4. Si editó el archivo de configuración del servidor 258, reinicie RaptorXML Server como servicio para que se aplique la nueva configuración. Pruebe la conexión otra vez para asegurarse de que el servicio está en ejecución y se puede acceder a él.

Nota: los errores de inicio del servidor, el archivo de configuración utilizado y los errores de licencia se registran en el registro del sistema. Por tanto, si tiene problemas con el servidor, consulte el registro del sistema[259].

Para obtener más información sobre HTTPS consulte el apartado Configuración HTTPS[263].

6.1.1.1 Iniciar el servidor

Temas de este apartado:

- Ubicación del ejecutable del servidor[256]
- Iniciar RaptorXML como servicio (Windows)[256]
- Iniciar Raptor XML como servicio (Linux)[256]
- Iniciar RaptorXML como servicio (macOS)[257]

Ubicación del ejecutable del servidor

El ejecutable de RaptorXML Server se instala por defecto en esta carpeta:

```
<CarpetaArchivosProgramas>\Altova\RaptorXMLServer2024\bin\RaptorXML.exe
```

Puede usar este ejecutable para iniciar RaptorXML Server como servicio.

Iniciar RaptorXML como servicio en Windows

Durante el proceso de instalación se registra RaptorXML Server como servicio en Windows, pero después debe **iniciarlo** como servicio. Tiene varias opciones:

- Con Altova ServiceController, disponible en la bandeja del sistema. Si el icono de Altova ServiceController no aparece en la bandeja del sistema, haga clic en el menú **Inicio** y seleccione **Todos los programas | Altova | Altova LicenseServer | Altova ServiceController**.
- Con la consola de gestión de servicios de Windows: **Panel de control | Todos los elementos de Panel de control | Herramientas administrativas | Servicios**.
- Iniciando el símbolo del sistema con derechos de administrador y usando este comando en cualquier directorio: **`net start "Altova RaptorXML Server"`**
- Con el ejecutable de RaptorXML Server en una ventana del símbolo del sistema: **`RaptorXMLServer.exe debug`**. Esto inicia el servidor y la información de actividad del servidor aparece directamente en la ventana del símbolo del sistema. Para ver/ocultar esta información, modifique la opción `http.log-screen`[259] del archivo de configuración del servidor[258]. Para detener el servidor, pulse **Ctrl+Interrumpir** (o **Ctrl+Pausa**). Cuando el servidor se inicia de esta forma, el servidor se detiene cuando se cierra la consola de la línea de comandos o cuando el usuario cierra sesión.

Iniciar RaptorXML como servicio en Linux

Utilice este comando para iniciar RaptorXML Server como servicio:

[< Debian 8]	**`sudo /etc/init.d/raptorxmlserver start`**
[≥ Debian 8]	**`sudo systemctl start raptorxmlserver`**
[< CentOS 7]	**`sudo initctl start raptorxmlserver`**

 © 2017-2023 Altova GmbH

[≥ CentOS 7]	sudo systemctl start raptorxmlserver
[< Ubuntu 15]	sudo initctl start raptorxmlserver
[≥ Ubuntu 15]	sudo systemctl start raptorxmlserver
[RedHat]	sudo initctl start raptorxmlserver

Si necesita detener RaptorXML Server, utilice este otro comando:

[< 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

Iniciar RaptorXML como servicio en macOS

Utilice este comando para iniciar RaptorXML Server como servicio:

```
sudo launchctl load /Library/LaunchDaemons/com.altova.RaptorXMLServer2024.plist
```

Si necesita detener RaptorXML Server, utilice este otro comando:

```
sudo launchctl unload /Library/LaunchDaemons/com.altova.RaptorXMLServer2024.plist
```

6.1.1.2 Probar la conexión

Temas de este apartado:

- Solicitud GET para probar la conexión 258
- Respuesta del servidor y ejemplo de datos JSON 258

Solicitud GET para probar la conexión

Tras iniciar RaptorXML Server puede probar la conexión con ayuda de una solicitud `GET`. También puede escribir la solicitud en la barra de dirección del explorador.

```
http://localhost:8087/v1/version
```

Nota: la interfaz y el número de puerto de RaptorXML Server se especifican en el archivo de configuración del servidor, `server_config.xml`, que se describe en el apartado Configurar el servidor 258.

Respuesta del servidor y ejemplo de datos JSON

Si el servicio está en ejecución y el servidor está configurado correctamente, la solicitud no debería dar ningún error. RaptorXML Server devuelve información sobre su número de versión en forma de datos JSON:

```
{
    "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"
}
```

Nota: si modifica la configuración del servidor (al editar el archivo de configuración del servidor 258), recomendamos que pruebe la conexión una vez más.

6.1.1.3 Configurar el servidor

Temas de este apartado:

- Archivo de configuración del servidor: configuración inicial 258
- Archivo de configuración del servidor: modificar configuración inicial, volver a la configuración inicial 258
- Archivo de configuración del servidor: ejemplo y opciones de configuración 259
- Archivo de configuración del servidor: descripción de las opciones de configuración 259
- Configurar la dirección del servidor 262

Archivo de configuración del servidor: configuración inicial

RaptorXML Server se configura por medio de un archivo de configuración llamado `server_config.xml`, que se coloca por defecto en esta carpeta:

```
C:\Program Files (x86)\Altova\RaptorXMLServer2024\etc\server_config.xml
```

La configuración inicial para RaptorXML Server define:

- El número de puerto `8087` como puerto del servidor
- Que el servidor solamente escucha a conexiones locales (`localhost`).
- Que el servidor escribe los resultados en la carpeta `C:\ProgramData\Altova\RaptorXMLServer2024\Output\`.

Las demás opciones de configuración predeterminadas aparecen más abajo en el fragmento 259 del archivo de configuración `server_config.xml`.

Archivo de configuración del servidor: modificar o volver a la configuración inicial

Si quiere cambiar la configuración inicial, edite el archivo de configuración del servidor `server_config.xml` (ver más abajo 259), guárdelo y después reinicie RaptorXML Server como servicio.

Si quiere recrear el archivo de configuración original (para volver a la configuración inicial), ejecute el comando `createconfig`:

```
RaptorXML.exe createconfig
```


© 2017-2023 Altova GmbH

Al ejecutar este comando, el archivo de configuración inicial se recrea y sobrescribe el archivo `server_config.xml`. Por tanto, el comando `createconfig` sirve para devolver al servidor a su configuración inicial de fábrica.

Archivo de configuración: opciones y ejemplo

A continuación puede ver el archivo de configuración del servidor `server_config.xml` en su versión original. Las opciones de configuración se explican más abajo.

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\RaptorXMLServer2024\output\</server.output-
root-dir>
  <server.script-root-dir>C:\Program
Files\Altova\RaptorXMLServer2024\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\RaptorXMLServer2024\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\RaptorXMLServer2024\Log\access.log</http.access-
    file>
    <http.error-file>C:\ProgramData\Altova\RaptorXMLServer2024\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\RaptorXMLServer2024\etc\cert\key.pem</https.private-key>
  <https.certificate>C:\Program
Files\Altova\RaptorXMLServer2024\etc\cert\cert.pem</https.certificate>
  <!--<https.certificate-chain>/path/to/chain.pem</https.certificate-chain>-->

</config>
```

Opciones de configuración

language

Define el idioma de los mensajes del servidor, en el elemento opcional `language`. El valor predeterminado es `es` (español) y los valores permitidos son `en|de|es|fr|ja` (inglés, alemán, español, francés y japonés respectivamente). Para aprender a localizar (=traducir) RaptorXML, consulte el apartado Comandos de localización[213].

server.unrestricted-filesystem-access

- Si el valor es `true` (valor predeterminado), los archivos de salida se escriben directamente en la ubicación especificada por el cliente y en los scripts Python (y posiblemente sobrescriben los archivos ya existentes del mismo nombre). Sin embargo, no se pueden utilizar rutas de acceso de archivo locales para acceder a archivos desde un equipo remoto por HTTP. Por tanto, si RaptorXML se está ejecutando en un equipo remoto, utilice el valor **`false`** para esta opción. El valor **`true`** solo se puede usar si el cliente y el servidor están en el mismo equipo y desea escribir los archivos de salida en un directorio de ese equipo.
- Si el valor es `false`, los archivos se escriben en el directorio del trabajo del directorio de salida[259] y los URI de estos archivos se incluye en el documento de resultados[281]. Por tanto, el valor `false` aporta mayor seguridad, porque los archivos solo se pueden escribir en el disco en un directorio del trabajo especializado y conocido. Los archivos de salida del trabajo se pueden copiar después en otras ubicaciones.

server.output-root-dir
Directorio en el que se guardan los documentos de salida de todos los trabajos.

server.script-root-dir
Directorio en el que se deben guardar los scripts Python[371] de confianza. Cuando se usa con la interfaz HTTP, la opción `script` solamente funciona si se utilizan scripts del directorio de confianza. Si especifica un script Python de otro directorio, se produce un error. Para más información consulte el apartado *Trabajar con scripts Python seguros*[371].

server.default-script-api-version
Versión predeterminada de la API de Python para ejecutar scripts de Python. Se usa por defecto la versión más reciente de la API. Los valores permitidos actualmente son 1 y 2.

server.catalog-file
La URL del archivo de catálogo XML que se debe usar. El archivo de catálogo `RootCatalog.xml` está por defecto en la carpeta `<CarpetaArchivosProgramas>\Altova\RaptorXMLServer2024\etc`. Solamente debe utilizar la opción `server.catalog-file` si quiere cambiar de archivo catálogo.

server.log-file
Nombre y ubicación del archivo de registro del servidor. Los eventos del servidor, como *Se inició/Se detuvo el servidor,* se registran continuamente en el registro de eventos del servidor y aparecen en un visor de eventos del sistema, como el visor de eventos de Windows. Además, los mensajes del registro se pueden escribir en el archivo indicado por medio de la opción `server.log-file`. El archivo de registro del servidor, por tanto, contiene información sobre todas las actividades del servidor, incluidos los errores de inicio del servidor, el archivo de configuración utilizado y los errores de licencia.

 © 2017-2023 Altova GmbH

http.enable
Un valor booleano para habilitar o deshabilitar HTTP: `true | false`. HTTP puede habilitarse/deshabilitarse independientemente de HTTPS y ambos pueden estar activos simultáneamente.

http.environment
Entornos internos de raptorxml: `production | development`. El entorno de desarrollo `development` está enfocado a los requisitos de los desarrolladores y permite realizar tareas de depuración con mayor facilidad que el entorno de producción `production`.

http.socket-host
Interfaz por la que se accede a RaptorXML Server. Si quiere que RaptorXML Server acepte conexiones desde equipos remotos, quite la marca de comentario de este elemento y defina su contenido como `0.0.0.0`, es decir: `<http.socket-host>0.0.0.0</http.socket-host>`. Esto almacena el servicio en cada interfaz direccionable del equipo servidor. En este caso, compruebe que las opciones del servidor de seguridad están configuradas correctamente. Las excepciones de entrada del servidor de seguridad para los productos de Altova deben registrarse de la siguiente manera:

- Altova LicenseServer: puerto `8088`
- Altova RaptorXML Server: puerto `8087`
- Altova FlowForce Server: puerto `8082`

http.socket-port
El puerto por el que se accede al servicio. El puerto debe ser fijo y conocido para que las solicitudes HTTP se puedan direccionar correctamente al servicio.

http.log-screen
Si inició RaptorXML Server con el comando `RaptorXMLServer.exe debug`, (*ver Iniciar el servidor* 256) y el valor de `http.log-screen` es `true`,
la actividad del servidor aparece en la consola de la línea de comandos. De lo contrario, la actividad del servidor no se muestra en la consola. Además de aparecer en pantalla, la actividad se registra en archivos de registro.

http.access-file
Nombre y ubicación del archivo de acceso HTTP. El archivo de acceso contiene información sobre la actividad relacionada con el acceso que puede ayudarle a resolver posibles problemas de conexión.

http.error-file
Nombre y ubicación del archivo de errores HTTP. El archivo de errores contiene errores relacionados con el tráfico entrante y saliente del servidor. Si hay problemas de conexión, este archivo puede ayudarle a resolverlos.

http.max_request_body_size
Esta opción especifica el tamaño máximo del cuerpo de la solicitud que RaptorXML Server debe aceptar. El valor predeterminado es 100MB. Si el sitio de un cuerpo de solicitud es mayor que el valor indicado en esta opción, el servidor responderá con el error HTTP 413: `Entidad de solicitud demasiado larga`. El valor de

la opción debe superior o igual a cero. El límite se puede deshabilitar si establece esta opción como `http.max_request_body_size=0`.

`https.enable`
Un valor booleano para habilitar o deshabilitar HTTPS: `true | false`. HTTPS puede habilitarse/deshabilitarse independientemente de HTTP y ambos pueden estar activos simultáneamente. La opción HTTPS está deshabilitada por defecto y debe habilitarse cambiando el valor de esta opción por **`true`**.

`https.socket-host`
Toma un valor de cadena que es la dirección de host donde se aceptan conexiones HTTP. Para aceptar conexiones del host local solamente, defina **`localhost`** o **`127.0.0.1`**. Si desea que RaptorXML Server acepte conexiones de todos los equipos remotos, defina el valor `0.0.0.0`, de la siguiente manera: `<http.socket-host>0.0.0.0</http.socket-host>`. Esto hospeda el servicio en todas las interfaces direccionables del equipo servidor. En este caso se debe comprobar que el firewall está configurado correctamente. Las excepciones de entrada del firewall para productos de Altova deben registrarse de la siguiente manera: Altova LicenseServer: puerto `8088`; Altova RaptorXML Server: puerto `8087`; Altova FlowForce Server: puerto `8082`. También puede usar direcciones IPv6 como: `'::'`.

`https.socket-port`
Un valor entero que es el puerto donde se acepta HTTPS. El puerto debe ser fijo y conocido para que las solicitudes HTTP se puedan dirigir correctamente al servicio.

`https.private-key, https.certificate`
Identificadores URI que son rutas de acceso de la clave privada del servidor y de los archivos de certificado del servidor respectivamente. Ambos son obligatorios. Consulte los apartados Configuración HTTPS 263 y Configurar el cifrado SSL 264 para obtener más información. En equipos Windows también puede usar rutas de acceso Windows.

`https.certificate-chain`
Opción opcional. Se trata de un URI que encuentra el archivo de certificado intermedio. Si tiene dos certificados intermedios (principal y secundario), combínelos en un solo archivo siguiendo las instrucciones del paso nº7 del apartado Configurar el cifrado SSL 264. Consulte los apartados Configuración HTTPS 263 y Configurar el cifrado SSL 264 para obtener más información

La dirección de **RaptorXML Server**

La dirección HTTP del servidor está formada por el host y el puerto del socket:

```
http://{socket-host}:{socket-puerto}/
```

La dirección de la configuración inicial es:

```
http://localhost:8087/
```

Para cambiar de dirección, cambie las opciones `http.socket-host` y `http.socket-port` del archivo de configuración del servidor `server_config.xml`. Por ejemplo, si el equipo

 © 2017-2023 Altova GmbH

servidor tiene la dirección IP `123.12.123.1` y se cambiaron las opciones de configuración por:

```
<http.socket-host>0.0.0.0</http.socket-host>
<http.socket-port>8087</http.socket-port>
```

Puede dirigirse a RaptorXML Server con:

```
http://123.12.123.1:8087/
```

Nota: tras modificar el archivo de configuración del servidor `server_config.xml`, es necesario reiniciar RaptorXML Server para que los cambios se apliquen.

Nota: si tiene problemas para conectarse con RaptorXML Server, puede que los archivos `http.access-file` y `http.error-file` le ayuden a resolver el problema.

Nota: los mensajes enviados a RaptorXML Server deben incluir nombres de ruta de acceso válidos en el equipo servidor. A los documentos del equipo servidor se puede acceder de forma local o remota (en el último caso mediante URI HTTP, por ejemplo).

6.1.1.4 Configuración HTTPS

RaptorXML Server permite el inicio como servidor HTTP y también como servidor HTTPS. Ambos tipos de conexión pueden estar activos de forma simultánea.

Habilitar HTTPS

La opción HTTPS está deshabilitada por defecto. Para habilitar HTTPS en el archivo de configuración del servidor 258, `server_config.xml`, defina el valor **`true`** para la opción **`https.enable`**. Modifique las distintas opciones HTTPS del archivo de configuración 258 en función de los requisitos de servidor.

Clave privada y certificado

Hay dos maneras de obtener una clave privada y archivos de certificado:

- De una entidad de certificación siguiendo los pasos descritos en el apartado Configurar el cifrado SSL 264.
- Creando un certificado autofirmado con este comando OpenSSL (modificado según el entorno que se utilice):

```
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"
```

Probar la conexión

Puede probar la conexión con ayuda de la herramienta de la línea de comandos curl que sirve para transferir datos con direcciones URL. Puede usar este comando:

```
curl.exe https://localhost:443/v1/version
```

Si el certificado no es de confianza, use la opción `-k`:

```
curl.exe -k https://localhost:443/v1/version
```

Este comando ejecuta el ejemplo HTTP Python que se distribuye con RaptorXML Server:

```
python3.exe examples\ServerAPI\python\RunRaptorXML.py --host localhost -p 443 -s
```

6.1.1.5 Configurar el cifrado SSL

Si desea cifrar la comunicación entre RaptorXML Server con el protocolo SSL, será necesario:

- Generar una clave privada SSL y crear un archivo de certificado de clave pública SSL
- Configurar RaptorXML Server para la comunicación con cifrado SSL.

Más abajo encontrará instrucciones para hacerlo.

Este método utiliza el kit de herramientas OpenSSL de código abierto para gestionar el cifrado SSL. Por tanto, los pasos que se describen en las instrucciones solo funcionarán en equipos con OpenSSL. El kit de herramientas OpenSSL suele estar instalado por defecto en la mayoría de las distribuciones de Linux y en equipos macOS, pero también se puede instalar en equipos Windows. En la wiki de OpenSSL encontrará enlaces para descargar a los binarios de instalación.

1. Generar una clave privada

   SSL requiere tener instalada una **clave privada** en RaptorXML Server. Esta clave privada se utilizará para cifrar todos los datos de RaptorXML Server. Para crear la clave privada utilice este comando de OpenSSL:

   ```
   openssl genrsa -out private.key 2048
   ```

   Esto crea un archivo llamado `private.key`, que contiene la clave privada. Recuerde dónde guarda el archivo porque lo necesitará para (i) generar la solicitud de firma de certificado (CSR) y (ii) instalarlo en RaptorXML Server.

2. Solicitudes de firma de certificado (CSR)

   La solicitud de firma de certificado (CSR) se envía a una entidad de certificación (como VeriSign o Thawte) para solicitar un certificado de clave pública. La CSR se basa en la clave privada y contiene información sobre su compañía. Cree una CSR con este comando de OpenSSL:

   ```
   openssl req -new -nodes -key private.key -out my.csr
   ```

   Este comando aporta el archivo de clave privada `private.key` creado en el paso nº1.

   Durante la generación de la CSR deberá indicar datos sobre su compañía. La entidad de certificación utilizará estos datos para verificar su identidad:

 © 2017-2023 Altova GmbH

- *País*
- *Localidad (ciudad donde está situada su compañía)*
- *Organización (nombre de su compañía).* No utilice caracteres especiales porque el certificado no será válido.
- *Nombre común (nombre DNS de su servidor).* Debe ser idéntico al nombre oficial de su servidor (es decir, debe ser el nombre DNS que utilizarán las aplicaciones cliente para conectarse al servidor).
- *Contraseña de comprobación.* Deje este campo vacío.

3. Comprar un certificado SSL

   Compre un certificado SSL de una entidad de certificación reconocida, como VeriSign o Thawte. En adelante utilizamos el procedimiento de VeriSign, pero es similar al de otras entidades de certificación:

   - Visite el sitio web de VeriSign.
   - Haga clic en **Buy SSL Certificates**.
   - Hay varios tipos de certificados SSL a la venta. Para RaptorXML Server es suficiente un certificado Secure Site o Secure Site Pro. Como no existe una barra de dirección verde no será necesaria una comprobación extendida (EV).
   - Siga los pasos y rellene el formulario de compra con sus datos.
   - Cuando se le solicite la CSR (creada en el paso nº2), copie y pegue el contenido del archivo **`my.csr`** en el formulario.
   - Efectúe el pago con una tarjeta de crédito válida.

   > **Tiempo de espera para obtener el certificado**
   > El certificado de una entidad de certificación SSL suele tardar **dos o tres días laborales**. Tenga esto en cuenta a la hora de configurar RaptorXML Server.

4. Recibir la clave pública de la entidad de certificación

   La autoridad de certificación elegida terminará el proceso de registro en dos o tres días laborales. Entre tanto es posible que reciba algún correo electrónico o llamada telefónica para comprobar si tiene autorización para solicitar un certificado SSL para su dominio DNS.

   Una vez completado el proceso de registro y autorización, recibirá un correo electrónico con la clave pública de su certificado SSL. Esta clave pública estará en texto sin formato o será un archivo **`.cer`**.

5. Guardar la clave pública en un archivo

   Para poder usarla con RaptorXML Server la clave pública debe estar guardada en un archivo **`.cer`**. Si recibió la clave pública como texto sin formato, copie y pegue todas las líneas de la clave, desde `--BEGIN CERTIFICATE--` hasta `--END CERTIFICATE--` en un archivo de texto que llamaremos **`miCertificado.cer`**.

6. Guardar los certificados intermedios de la autoridad de certificación en un archivo

Para completar el certificado SSL necesitará otros dos certificados: el **certificado intermedio principal** y el **certificado intermedio secundario**. En el sitio web de su autoridad de certificación encontrará el contenido de los certificados intermedios:

- Certificados intermedios de Verisign: https://knowledge.verisign.com/support/ssl-certificates-support/index?page=content&id=AR657&actp=LIST&viewlocale=en_US
- Certificados intermedios de Verisign para su producto Secure Site: https://knowledge.verisign.com/support/ssl-certificates-support/index?page=content&id=AR1735

Copie y pegue los dos certificados intermedios en sendos archivos de texto y guárdelos en el equipo.

7. (Opcional) Combinar los certificados en un solo archivo de certificado de clave pública

   Ahora cuenta con tres archivos de certificado:

   - La clave pública (`miCertificado.cer`)
   - El certificado intermedio secundario
   - El certificado intermedio principal

   Si lo desea, puede integrar los certificados intermedios en el certificado de clave pública. Esto se explica a continuación. Si lo prefiere, puede usar la opción del archivo de configuración 259 **`https.certificate-chain`** para especificar la ubicación de los certificados intermedios.

   Todos contienen bloques de texto entre líneas similares a estas:

   ```
   --BEGIN CERTIFICATE--
     ...
   --END CERTIFICATE--
   ```

   Ahora copie y pegue los tres certificados en un solo archivo, uno detrás del otro. El orden de aparición es importante: (i) primero la clave pública, (ii) después el certificado intermedio secundario y (iii) por último el certificado intermedio principal. Compruebe que no hay líneas vacías entre un certificado y el siguiente.

   ```
   --BEGIN CERTIFICATE--
     clave pública de miCertificado.cer (paso nº5)
   --END CERTIFICATE--
   --BEGIN CERTIFICATE--
     certificado intermedio secundario (paso nº6)
   --END CERTIFICATE--
   --BEGIN CERTIFICATE--
     certificado intermedio principal (paso nº6)
   --END CERTIFICATE--
   ```

   Guarde el texto resultante en un archivo llamado **`publickey.cer`**, que es ya el certificado de clave pública de su certificado SSL. Incluye el certificado de clave pública y la cadena de confianza (es decir, los certificados intermedios utilizados por la entidad de certificación para firmar el certificado.

 © 2017-2023 Altova GmbH

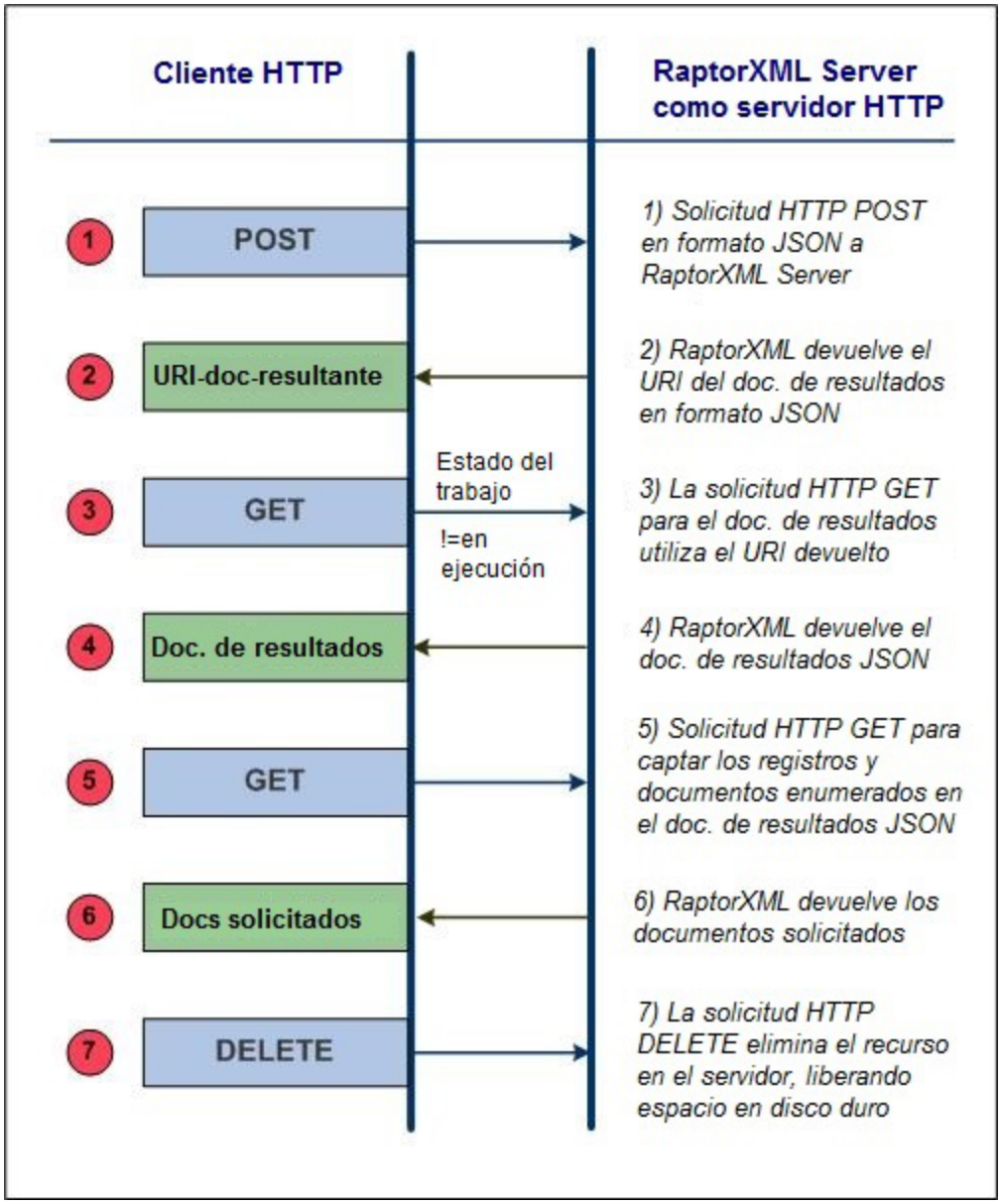
6.1.2 Solicitudes cliente

Tras iniciar RaptorXML Server como servicio 256, podrá acceder a sus funciones cualquier cliente HTTP que pueda:

- usar los métodos HTTP `GET`, `PUT`, `POST` y `DELETE`
- establecer el campo de encabezado `Content-Type`

> **Un sencillo cliente HTTP**
>
> Hay varios clientes web que puede descargar de Internet, pero el cliente web RESTClient de Mozilla es sencillo y fiable y se puede añadir como complemento de Firefox. Es fácil de instalar, es compatible con los métodos HTTP necesarios para trabajar con RaptorXML y ofrece funciones de color de sintaxis JSON. Si no tiene experiencia con clientes HTTP, recomendamos trabajar con RESTClient. No obstante, la instalación y el uso del cliente RESTClient corre por su cuenta y riesgo.

Una solicitud cliente normal suele estar formada por varios pasos, como muestra el diagrama siguiente.



A continuación destacamos los aspectos más importantes de cada paso, señalando en **negrita** los términos más importantes.

1. Los métodos HTTP `POST` se utilizan para realizar una solicitud[269] y el cuerpo de la solicitud está en **formato JSON**. La solicitud puede ser para cualquier función de RaptorXML Server. Por ejemplo, la solicitud puede ser para una validación o una transformación XSLT. Los comandos, los argumentos y las opciones utilizados en la solicitud son los mismos que los de la línea de comandos[58]. La solicitud se envía a `http://localhost:8087/v1/queue`, si suponemos que `localhost:8087` es la dirección de RaptorXML Server (la dirección inicial del servidor[262]). Dicha solicitud recibe el nombre de **trabajo de RaptorXML Server**.

 © 2017-2023 Altova GmbH

2. Si RaptorXML Server recibe y acepta procesar la solicitud, después de procesar el trabajo se genera un **documento de resultados** que contiene los resultados de la acción del servidor. El **URI de este documento de resultados** (ver *URI-doc-resultante* en el diagrama anterior), se devuelve al cliente 281. Recuerde que el URI se devolverá inmediatamente después de que se acepte procesar el trabajo (después de que se ponga en la cola) e incluso si el procesamiento no ha terminado.

3. El cliente envía al servidor una solicitud para el documento de resultados 281 (usando el URI del documento de resultados) en un método `GET`. Si todavía no empezó el procesamiento del trabajo o si, cuando se recibe la solicitud, el procesamiento continúa, el servidor devuelve el estado *En ejecución*. La solicitud `GET` debe repetirse hasta que el procesamiento del trabajo haya finalizado y se haya creado el documento de resultados.

4. RaptorXML Server devuelve el documento de resultados en formato JSON 282. El documento de resultados puede contener los **URI de los documentos de errores o de salida** generados por RaptorXML Server al procesar la solicitud inicial. Por ejemplo, si una validación devuelve errores, se devuelve un registro de errores. Los documentos de salida principales, como el resultado de la transformación XSLT, se devuelven si el trabajo generador de resultados finaliza correctamente.

5. El cliente envía al servidor los URI de los documentos de salida 285 recibidos en el paso nº4 a través de un método HTTP `GET`. Cada solicitud se envía en un método `GET` distinto.

6. RaptorXML Server devuelve los documentos solicitados 285 en respuesta a las solicitudes `GET` realizadas en el paso nº5.

7. El cliente puede eliminar del servidor documentos no deseados 286 generados como resultado de una solicitud de trabajo. Esto se hace enviando en un método HTTP `DELETE` el URI del documento de resultados en cuestión. Se eliminan del disco todos los archivos relacionados con dicho trabajo. Esto incluye el documento de resultados, los archivos temporales, los documentos de errores y los documentos de salida. Este paso es muy práctico si quiere liberar espacio en el disco duro del servidor.

Todos estos pasos se describen con más detalle en los apartados de esta sección.

6.1.2.1 Iniciar trabajos con POST

Temas de este apartado:

- Enviar la solicitud 269
- Sintaxis JSON para solicitudes POST 269
- Cargar archivos con la solicitud POST 271
- Cargar archivos ZIP 272

Enviar la solicitud

Los trabajos de RaptorXML Server se inician con el método HTTP `POST`.

Método HTTP	URI	Campo de encabezado Content-Type	Cuerpo

POST	http://localhost:8087/v1/queue/	application/json	JSON

Observe que:

- El URI de la tabla tiene una dirección de servidor que utiliza las opciones de la configuración inicial[258].
- El URI tiene una ruta de acceso `/v1/queue/`, que debe estar presente en el URI. Puede entenderse como una carpeta abstracta en memoria y en ella se coloca el trabajo.
- El número de versión `/vN` es el que devuelve el servidor (no necesariamente el que aparece en esta documentación). El número que devuelve el servidor es el número de versión de la interfaz HTTP actual. Los números de versión anteriores indican versiones antiguas de la interfaz HTTP y se admiten para garantizar la compatibilidad con versiones anteriores.
- El encabezado debe incluir el campo: `Content-Type: application/json`. Sin embargo, si quiere cargar archivos dentro del cuerpo de la solicitud `POST`, el encabezado del mensaje debe tener el tipo de contenido `multipart/form-data` (es decir, `Content-Type: multipart/form-data`). Para más información consulte el apartado Cargar archivos con la solicitud POST[271].
- El cuerpo de la solicitud debe estar en formato JSON.
- Los archivos que desea procesar deben estar en el servidor. Por tanto, debe copiar los archivos al servidor antes de realizar la solicitud o cargarlos junto con la solicitud POST[271]. En este último caso, el encabezado del mensaje debe tener el tipo de contenido `multipart/form-data`. Para más información consulte el apartado Cargar archivos con la solicitud POST[271].

Esta sería la solicitud en formato JSON para comprobar si un archivo XML tiene un formato correcto:

```
{
    "command": "wfxml", "args": [ "file:///c:/Test/Report.xml" ]
}
```

Los comandos, sus argumentos y sus opciones se documentan en la sección Interfaz de la línea de comandos[58].

Sintaxis JSON para solicitudes HTTP `POST`

```
{
   "command": "Nombre-Comando",
   "options": {"opción1": "opción1-valor", "opción2": "opción2-valor"},
   "args"   : ["file:///c:/nombreArchivo1", "file:///c:/nombreArchivo2"]
}
```

- El texto de color negro es obligatorio y debe incluirse. Esto incluye todas las llaves, comillas, dos puntos, comas y corchetes. Los espacios en blanco se pueden normalizar.
- El texto de color azul en cursiva son marcadores de posición para nombres de comandos, opciones, valores de opciones y valores de argumentos. Para más información consulte la sección Interfaz de la línea de comandos[58].

- Las claves `command` y `args` son obligatorias. La opción `options` es opcional. Algunas claves `options` tienen valores predeterminados. Por tanto, de todas estas opciones, solamente debe especificar aquellas cuyos valores predeterminados desee cambiar.

© 2017-2023 Altova GmbH

- Todas las cadenas de texto deben ir entre comillas dobles. Los valores booleanos y los números no necesitan ir entre comillas. El uso correcto sería {"error-limit": "unlimited"} y {"error-limit": 1}.
- Es recomendable usar los URI de archivo y no sus rutas de acceso. Recuerde que los URI usan barras diagonales, mientras que las rutas de acceso usan barras diagonales inversas. Además, en Windows es necesario añadir caracteres de escape a las barras diagonales inversas de las rutas de acceso (es decir, "c:\\dir\\nombreArchivo"). Por último, recuerde que los URI y las rutas de acceso son cadenas y, por tanto, deben ir entre comillas.

A continuación puede ver un ejemplo con opciones. Observe que algunas opciones (como `input` o `xslt-version`) toman un valor directo, mientras que otros (como `param`) toman un par clave-valor y, por tanto, necesitan otra sintaxis.

```
{
   "command": "xslt",
   "args": [
           "file:///C:/Work/Prueba.xslt"
           ],
   "options": {
              "input": "file:///C:/Work/Prueba.xml",
              "xslt-version": 1,
              "param": {
                       "key": "miPruebaParam",
                       "value": "ValorDeParám"
                       },
              "output": "file:///C:/temp/salida2.xml"
              }
}
```

El ejemplo que aparece a continuación muestra otro tipo de opción más: una matriz de valores (como en la opción `xsd` del ejemplo). En casos así, debe usar la sintaxis de una matrix JSON.

```
{
   "command": "xsi",
   "args": [
           "file:///C:/Work/Prueba.xml"
           ],
   "options": {
              "xsd" : ["file:///C:/Work/File1.xsd", "file:///C:/Work/Archivo2.xsd"]
              }
}
```

Cargar archivos con la solicitud POST

Puede cargar los archivos que deben procesarse dentro del cuerpo de la solicitud `POST`. En este caso, la solicitud `POST` debe crearse como se muestra a continuación.

Encabezado de la solicitud
En el encabezado de la solicitud, defina `Content-Type: multipart/form-data` y especifique una cadena de texto cualquiera como frontera. Por ejemplo:

```
Content-Type: multipart/form-data; boundary=---FronteraParcial
```

El objetivo de la frontera es delimitar las diferentes partes `form-data` del cuerpo de la solicitud (*ver más abajo*).

Cuerpo de la solicitud: parte del mensaje
El cuerpo de la solicitud está formado por varias partes `form-data`, separadas por la cadena de frontera especificada en el encabezado (*ver más arriba*):

- *Partes `form-data` obligatorias:* la parte **`msg`**, que especifica la acción de procesamiento solicitada, y la parte **`args`**, que contiene los archivos que deben cargarse como argumentos del comando especificado en la parte **`msg`**. (*Ver ejemplo que aparece más abajo*).
- *Parte `form-data` opcional:* la parte **`additional-files`** contiene los archivos a los que se hace referencia desde las partes `form-data` obligatorias **`msg`** y **`args`**. Además, las partes `form-data` con nombre de opción de comando también pueden incluir archivos para cargar.

Nota: todos los archivos cargados se crean en el mismo directorio virtual.

Consulte el Ejemplo nº1 (con llamadas): validar XML 273 para ver un ejemplo de código y el Ejemplo nº2: usar un catálogo para buscar el esquema 274.

Pruebas con CURL

Puede usar aplicaciones externas de transferencia de datos como CURL (http://curl.haxx.se/) para probar la solicitud POST. CURL ofrece una opción de seguimiento muy práctica que genera y enumera las fronteras parciales de las solicitudes. Esto evita tener que crear a mano las fronteras parciales. Para más información consulte el apartado Pruebas con CURL 277.

Cargar archivos ZIP

También puede cargar archivos ZIP y hacer referencia a los ficheros del archivo ZIP con el esquema **`additional-files`**. Por ejemplo:

```
additional-files:///migranarchivo.zip%7Czip/instanciagrande.xml
```

Nota: en la parte **`|zip/`** el URI tiene que estar entre caracteres de escape (**`%7Czip/`**) para ajustarse al RFC del URI porque el símbolo **`|`** no está permitido. El uso de patrones glob (**`*`** y **`?`**) también está permitido. Por tanto, para validar todos los archivos XML del archivo ZIP puede usar algo así:

```
{"command": "xsi", "args": ["additional-files:///migranarchivo.zip%7Czip/*.xml"],
"options": {...}}
```

 © 2017-2023 Altova GmbH

Para ver otro ejemplo de código consulte el Ejemplo nº3: usar archivos ZIP 275.

6.1.2.1.1 Ejemplo nº1 (con llamadas): validar XML

A continuación puede ver un ejemplo en el que aparece el cuerpo de una solicitud `POST` seguida de una tabla donde se explican todas sus partes (que aparecen numeradas). El comando de la ILC equivalente al comando enviado con esta solicitud sería:

```
raptorxml xsi Primero.xml Segundo.xml --xsd=Demo.xsd
```

La solicitud pide validar dos archivos XML con un esquema. El cuerpo de la solicitud tendría este aspecto, si imaginamos que en el encabezado se especificó la cadena de frontera `---PartBoundary` (*ver* *Encabezado de solicitud* 271).

```
-----PartBoundary                                                                    1
Content-Disposition: form-data; name="msg"
Content-Type: application/json

{"command": "xsi", "options": {} , "args": []}                                       2

-----PartBoundary                                                                    3
Content-Disposition: attachment; filename="Primero.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="Segundo.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 El nombre de las fronteras de las principales partes `form-data` se declaran en el encabezado de la solicitud[271]. El separador de frontera de parte debe ser una cadena única que no aparezca en ninguna otra parte de los documentos incrustados. Se le añade un prefijo de dos guiones y se utiliza para separar sus partes. La primera parte `form-data` (de este ejemplo) es la parte **`msg`**. Observe que su tipo de contenido es `application/json`.

2 Esta es la sintaxis estándar para solicitudes HTTP [270] `POST` [270]. Si `args` contiene una referencia a un archivo y se cargaron más archivos, ambos grupos de archivos se pasan al servidor.

3 El primer miembro de la matriz `args` es un archivo adjunto llamado `Primero.xml`.

4 El texto del archivo `Primero.xml`. Contiene una referencia a un esquema llamado `Demo.xsd`, que también se cargará (en la parte `form-data` **`additional_files`**).

5 El segundo miembro de la matriz **`args`** es un archivo adjunto llamado `Segundo.xml`.

6 El texto del archivo `Segundo.xml`. También contiene una referencia al esquema `Demo.xsd` (*ver el punto 7*).

7 El final del último miembro de la matriz **`args`** (y por tanto de la matriz propiamente dicha) se indica añadiendo el sufijo `'--'` a la cadena de frontera de **`args`**.

8 El texto del archivo `Demo.xsd`.

9 El final de la parte **`additional_files`**. Observe que el último separador de frontera de parte tiene tanto un prefijo como un sufijo formados por dos guiones.

6.1.2.1.2 Ejemplo nº2: usar un catálogo para buscar el esquema

En este ejemplo se utiliza un archivo de catálogo para buscar el esquema XML al que hacen referencia los archivos XML que se deben validar.

```
-----PartBoundary
Content-Disposition: form-data; name="msg"
Content-Type: application/json

{"command": "xsi", "args": ["additional-files:///Primero.xml", "additional-
files:///Segundo.xml"], "options": {"user-catalog": "additional-files:///catalog.xml"}}

-----PartBoundary
Content-Disposition: attachment; filename="Primero.xml"; name="additional-files"
Content-Type: application/octet-stream

<?xml version="1.0" encoding="UTF-8"?>
<test xsi:noNamespaceSchemaLocation="http://ejemplo.com/Demo.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">42</test>
```

 © 2017-2023 Altova GmbH

```
-----PartBoundary
Content-Disposition: attachment; filename="Segundo.xml"; name="additional-files"
Content-Type: application/octet-stream

<?xml version="1.0" encoding="UTF-8"?>
<test xsi:noNamespaceSchemaLocation="http://ejemplo.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://ejemplo.com/Demo.xsd" uri="additional-
files:///Demo.xsd"/>
</catalog>

-----PartBoundary--
```

6.1.2.1.3 Ejemplo nº3: usar archivos ZIP

También puede cargar archivos ZIP y hacer referencia a los ficheros del archivo ZIP con el esquema `additional-files`. Por ejemplo:

```
additional-files:///migranarchivo.zip%7Czip/instanciagrande.xml
```

Nota: en la parte `|zip/` el URI tiene que estar entre caracteres de escape (`%7Czip/`) para ajustarse al RFC del URI porque el símbolo `|` no está permitido. El uso de patrones glob (`*` y `?`) también está permitido. Por tanto, para validar todos los archivos XML del archivo ZIP puede usar algo así:

```
{"command": "xsi", "args": ["additional-files:///migranarchivo.zip%7Czip/*.xml"],
"options": {...}}
```

Nota: `'Content-Disposition: form-data'` también es válido, junto con `'Content-Disposition: attachment'`. Como varias herramientas generan `form-data` como `content-disposition`, también se acepta el valor `form-data`.

Ejemplo: validar todos los archivos XML del archivo ZIP

En este ejemplo, se supone que todas las referencias de esquema son rutas de acceso relativas y que todos los esquemas están dentro del archivo ZIP.

```
-----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

Contenido binario del archivo Demo.zip

-----PartBoundary--
```

Ejemplo: validar archivos XML del archivo ZIP con referencias a esquemas externos

En este ejemplo los archivos XML del archivo ZIP se validan por medio de referencias a un esquema externo, que está en otro archivo ZIP.

```
-----PartBoundary
Content-Disposition: form-data; name="msg"
Content-Type: application/json

{"command": "xsi", "args": ["additional-files:///Instancias.zip%7Czip/*.xml"], "options":
{"user-catalog": "additional-files:///Schemas.zip%7Czip/catalog.xml"}}

-----PartBoundary
Content-Disposition: attachment; filename="Instancias.zip"; name="additional-files"
Content-Type: application/octet-stream

Contenido binario del archivo Instancias.zip

-----PartBoundary
Content-Disposition: attachment; filename="Esquemas.zip"; name="additional-files"
Content-Type: application/octet-stream

Contenido binario del archivo Esquemas.zip

-----PartBoundary--
```

 © 2017-2023 Altova GmbH

6.1.2.1.4 Pruebas con CURL

La aplicación CURL (http://curl.haxx.se/) es una utilidad de la línea de comandos que puede utilizarse para probar la solicitud POST. CURL ofrece una opción de seguimiento muy práctica que genera y enumera las fronteras de parte de las solicitudes. Puede utilizar estas fronteras en sus solicitudes directamente o utilizarlas como referencia.

Más abajo puede ver un ejemplo donde se valida un archivo XML con un esquema XML.

- En este ejemplo se supone que los comandos se ejecutan desde la carpeta donde residen los archivos que se deben validar (esto nos permite escribir rutas de acceso relativas sencillas para estos archivos). Si instaló la aplicación Altova XMLSpy, encontrará los archivos utilizados en el ejemplo en la carpeta `Examples` de la aplicación, cuya ubicación predeterminada es: `C:\Users\<nombreUsuario>\Documents\Altova\XMLSpy2024\Examples`.
- En este ejemplo se supone que RaptorXML Server se ejecuta localmente en el puerto 8087.

Para más información sobre las opciones de la línea de comandos CURL consulte la documentación de esta aplicación.

Llamar a CURL con el comando de validación en 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
```

Nota: en powershell, si utiliza comillas dentro de otras comillas, éstas deben ser de dos tipos diferentes (sencillas y dobles).

```
[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"}
```

Usar la URL de "result" para recuperar el resultado

```
[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":{}}]}
```


Opción de seguimiento de CURL

CURL ofrece la opción de seguimiento (`--trace-ascii`), que sigue el tráfico HTTP entrante y saliente del servidor. La opción es muy práctica porque enumera las fronteras de parte que son necesarias para iniciar trabajos con POST. Puede usar la información de seguimiento directamente o como referencia para crear fronteras de partes. En el fragmento de código que aparece a continuación puede ver la información de seguimiento que se obtiene al ejecutar el comando.

Fragmento de código de seguimiento

```
== 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
```

 © 2017-2023 Altova GmbH

```
<= 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
```

Nota: observe que también es válido usar `'Content-Disposition: form-data'` junto con `'Content-Disposition: attachment'`.

Llamar a CURL con el comando well-formed-check en 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
```

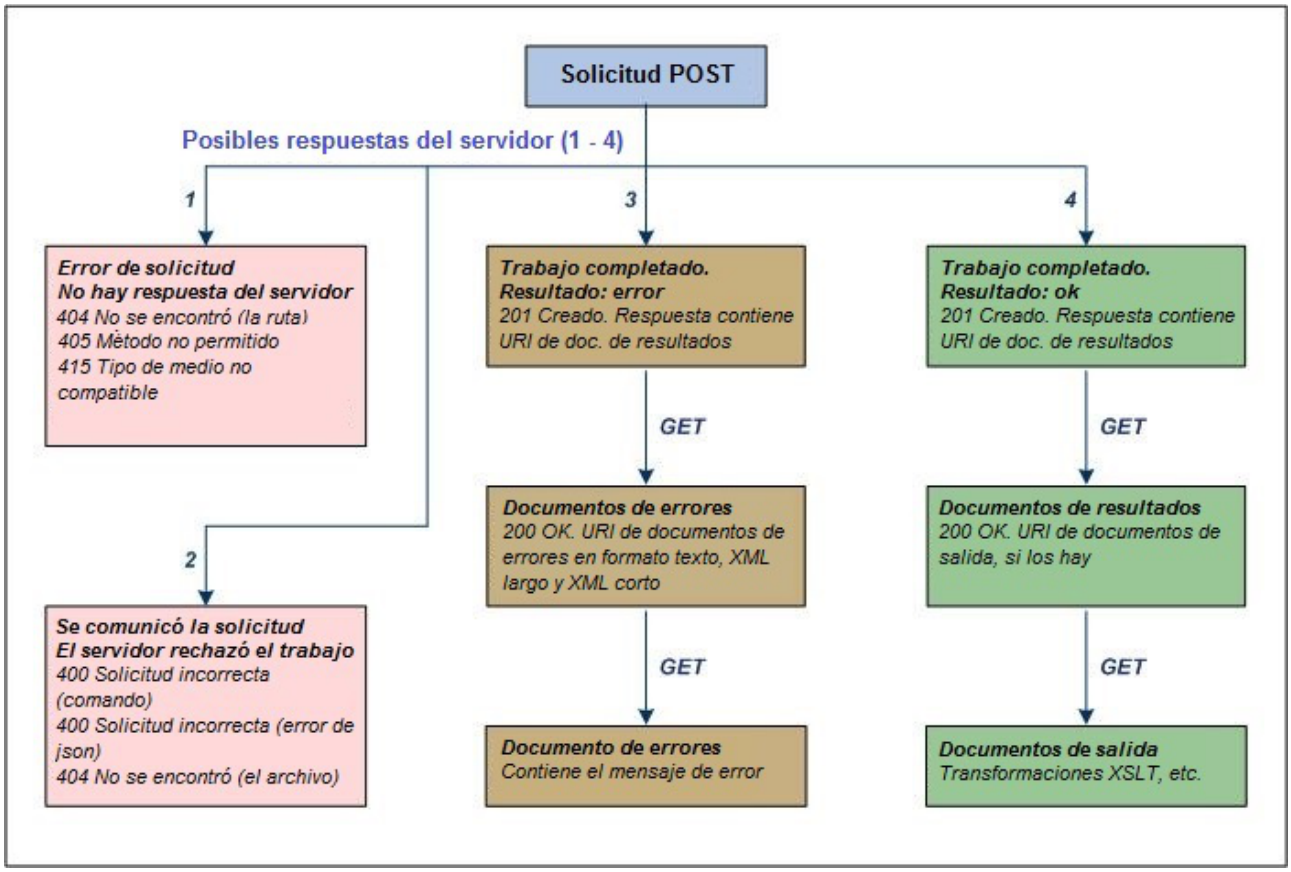
6.1.2.2 Respuesta del servidor a solicitudes POST

Temas de este apartado:

- Posibles respuestas del servidor 279
- Respuesta: Error de solicitud, sin respuesta del servidor 280
- Respuesta: Se comunicó la solicitud, pero el servidor rechazó el trabajo 280
- Respuesta: Se ejecutó el trabajo (con resultados positivos o negativos) 281

Cuando la solicitud POST se realiza correctamente, el trabajo se coloca en la cola de trabajos del servidor y se devuelve un mensaje `201 Creado` y el URI del documento de resultados. El trabajo se procesará lo antes posible. Mientras tato, si se solicita el documento de resultados 281, se devuelve el mensaje `"estado": "En ejecución"` (si el trabajo está sin completar) o el mensaje `"estado": "Distribuido"` (si el trabajo está en la cola del servidor pero todavía no se inició).

El resultado del trabajo (por ejemplo, de un trabajo de validación) puede ser negativo ("***Failed***") o positivo ("***OK***"). En ambos casos se devuelve el mensaje `201 Creado` y se genera el documento de resultados. También puede ocurrir que la solicitud POST no se comunique al servidor (*Error de solicitud*) o que se comunique al servidor pero este la rechace (Se *comunicó la solicitud, pero el servidor rechazó el trabajo*). Todas las posibles respuestas del servidor se resumen en este diagrama:



Error de solicitud, no hay respuesta del servidor

Cuando la solicitud se envía al servidor, los errores más comunes son:

Mensaje	Motivo
404 No se encontró (*la ruta*)	La ruta de acceso correcta es `http://localhost:8087/v1/queue/`
405 Método no permitido	El método especificado para este recurso no es válido. Utilice el método `POST`.
415 Tipo de medio no compatible	El encabezado del mensaje debería ser `Content-Type:application/json`.

Se comunicó la solicitud, pero el servidor rechazó el trabajo

Cuando la solicitud se envía al servidor, este puede rechazarla por varios motivos:

Mensaje	Motivo

© 2017-2023 Altova GmbH

400 Solicitud incorrecta (*comando malo*)	El comando de RaptorXML 58 no es correcto.
400 Solicitud incorrecta (*error de JSON*)	Hay un error de sintaxis JSON 270 en el cuerpo de la solicitud.
404 No se encontró (*el archivo*)	Revise la sintaxis del URI (o ruta) de archivo 270 de todos los archivos nombrados en el comando.

El trabajo se ejecutó (con resultados negativos o positivos)

Cuando se ejecuta un trabajo (por ejemplo, un trabajo de validación) su resultado puede ser positivo (***OK***) o negativo (***Failed***). Por ejemplo, el resultado de un trabajo de validación es positivo (***OK***) cuando el documento es válido y negativo (***Failed***) cuando el documento no es válido.

En ambos casos se ejecuta el trabajo, pero los resultados obtenidos son diferentes. En ambos casos se devuelve el mensaje `201 Creado` en cuanto el trabajo se coloca en la cola de trabajos del servidor. Además, en ambos casos se devuelve un URI de documento de resultados al cliente HTTP que hizo la solicitud. (Puede que el documento de resultados propiamente dicho no exista aún si el procesamiento del trabajo no comenzó/finalizó). Una vez creado el documento de resultados, puede recuperarlo con una solicitud HTTP `GET`. Además del documento de resultados se generan algunos documentos más:

- *Trabajo ejecutado. Resultado 'Failed':* se crea un registro de errores en tres formatos, texto, XML largo y XML corto. Los URI de estos tres documentos se envían en el documento de resultados (que está en formato JSON). Los URI se pueden usar en una solicitud HTTP `GET` para recuperar los documentos de errores 285.
- *Trabajo ejecutado. Resultado 'OK':* el trabajo se procesa correctamente y se crean documentos de salida (por ejemplo, los de una transformación XSLT). Si se generan archivos de salida, sus URI se incluyen en el documento de resultados. Estos URI se pueden usar en una solicitud HTTP `GET` para recuperar los documentos. Recuerde que no todos los trabajos tienen archivos de salida (por ejemplo, los trabajos de validación). Además, un trabajo puede terminar con un resultado positivo pero generar advertencias y otros mensajes en los archivos de errores. En este caso, los URI de los archivos de errores también se envían en el documento de resultados.

Para más información sobre estos documentos y cómo acceder a ellos consulte los apartados Obtener el documento de resultados 281 y Obtener documentos de errores y de salida 285.

6.1.2.3 Obtener el documento de resultados

Temas de este apartado:

- El URI del documento de resultados 282
- Recuperar el documento de resultados 282
  - Documento de resultados con los URI de los documentos de errores 282
  - Documento de resultados con los URI de los documentos de salida 283
  - Documento de resultados sin URI 284
- Acceder a los documentos de salida y de errores enumerados en el documento de resultados 285

El URI del documento de resultados

Cada vez que se crea un trabajo se genera un documento de resultados, independientemente de si el resultado del trabajo (p. ej. un trabajo de validación) es positivo (el documento es válido) o negativo (el documento no es válido). En ambos casos se devuelve un mensaje `201 Creado`. Este mensaje está en formato JSON y contiene el URI relativo del documento de resultados. Por ejemplo:

```
{
   "result": "/v1/results/E6C4262D-8ADB-49CB-8693-990DF79EABEB",
   "jobid": "E6C4262D-8ADB-49CB-8693-990DF79EABEB"
}
```

El objeto `result` contiene el URI relativo del documento de resultados. El URI es relativo a la dirección del servidor[262]. Por ejemplo, si la dirección del servidor es `http://localhost:8087/` (la dirección según la configuración inicial[258]), el URI ampliado del documento de resultados anterior sería:

```
http://localhost:8087/v1/results/E6C4262D-8ADB-49CB-8693-990DF79EABEB
```

Nota: el número de versión `/vN` es el que devuelve el servidor (no necesariamente el de esta documentación). El número que devuelve el servidor es el número de versión de la interfaz HTTP actual. Los números de versión antiguos indican versiones previas de la interfaz HTTP, que se admiten para garantizar la compatibilidad con versiones previas.

Recuperar el documento de resultados

Para obtener el documento de resultados puede enviar el URI ampliado del documento (*ver más arriba*[282]) en una solicitud HTTP `GET`. Se devuelve el documento de resultados, que puede ser de tres tipos (*ver más abajo*).

Nota: si el trabajo se coloca en la cola del servidor, este devuelve el URI del documento de resultados. Si el cliente solicita el resultado antes de que comience el trabajo (si todavía está en la cola), se devuelve el mensaje `"estado": "Distribuido"`. Si el trabajo se inició pero no terminó todavía (porque es un trabajo grande, por ejemplo), se devuelve el mensaje `"estado": "En ejecución"`. En ambos casos el cliente debe esperar un poco para volver a solicitar el documento de resultados.

Nota: en los ejemplos que aparecen a continuación se entiende que hay acceso de cliente restringido[255]. Es decir, se entiende que los documentos de errores, mensajes y de salida se guardan en el directorio correspondiente del servidor. Los URI de estos documentos en el documento de resultados son, por tanto, URI relativos. Ninguno de ellos es un URI de archivo (como el que se generaría con acceso de cliente no restringido[254]). Para más información consulte el siguiente apartado de esta sección.

Documento de resultados con los URI de los documentos de errores

Si el trabajo solicitado termina con el estado **`Failed`**, el trabajo devuelve un resultado negativo. Por ejemplo, un trabajo de validación devuelve un resultado de documento no válido. Los errores detectados durante la ejecución del trabajo se almacenan en registros de errores, creados en tres formatos de archivo: (i) texto, (ii) XML largo (registro detallado) y (iii) XML corto (registro menos detallado).

```
{
   "jobid": "6B4EE31B-FAC9-4834-B50A-582FABF47B58",
   "state": "Failed",
```

 © 2017-2023 Altova GmbH

```
    "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"
        }
      }
    ]
}
```


Es importante destacar que:

- Los trabajos tienen trabajos subordinados.
- Los errores de los trabajos subordinados se propagan al trabajo de nivel superior. Es decir, el estado del trabajo solamente puede ser OK si el de los trabajos subordinados también lo es.
- Cada trabajo y trabajo subordinado tiene su registro de errores.
- Los registros de errores incluyen advertencias. Es decir, incluso si el trabajo tiene el estado OK, puede haber archivos de errores.
- Los URI de los archivos de errores son relativos a la dirección del servidor (*ver más arriba* 282).

Documento de resultados con los URI de los documentos de salida

Si el trabajo solicitado termina con el estado OK, el trabajo devuelve un resultado positivo. Por ejemplo, un trabajo de validación devuelve un resultado de documento válido. Si el trabajo generó un documento de salida (por ejemplo, como resultado de una transformación XSLT), se devuelve el URI del documento de salida.


```
{
    "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":
       {
       }
    }
  ]
}
```

Es importante destacar que:

- El archivo de salida se crea en la carpeta `output` del trabajo. Puede usar su URI relativo para acceder al archivo.
- Los URI de los archivos de salida son relativos a la dirección del servidor (*ver más arriba* 282).
- El elemento `output-mapping` asigna el documento de salida del directorio del trabajo del servidor a la ubicación de archivo especificada por el cliente en la solicitud. Recuerde que solo tienen una asignación los documentos de salida especificados por el cliente en la solicitud. Los archivos relacionados con el trabajo generados por el servidor (como los archivos de errores) no tienen ninguna asignación.
- Otra opción es recuperar todos los documentos de resultados generados para un trabajo específico en un archivo ZIP por medio de la URL `"/v1/results/`*JOBID*`/output/zip"`. Esta característica no está disponible en el modo unrestricted filesystem. Recuerde que el archivo ZIP incluirá nombres de archivo alterados, que deberán asignarse a los nombres reales con ayuda del objeto `output-mapping`.

Documento de resultados sin URI

Si el trabajo solicitado terminó con el estado `OK`, significa que el trabajo devolvió un resultado positivo. Por ejemplo, un trabajo de validación devuelve un resultado de documento válido. Algunos trabajos, como los de validación o de comprobación de formato, no generan documentos de salida. Si un trabajo de este tipo termina con el estado `OK`, el documento de resultados generado no tendrá ni el URI de documento de salida ni el URI del registro de errores.

```
{
   "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":
       {
       }
     }
   ]
}
```

Observe que:

- Tanto los componentes de salida y de error del trabajo subordinado están vacíos en el ejemplo anterior.
- Un trabajo puede terminar con un estado OK, pero con advertencias y otro tipo de mensajes, que se registran en los archivos de errores. En este caso, el documento de resultados incluirá el URI de los archivos de errores, aunque el trabajo terminara con el estado OK.

Acceder a los documentos de errores y de salida enumerados en el documento de resultados

Puede acceder a los documentos de errores y de salida con solicitudes HTTP `GET`. Para más información consulte el siguiente apartado de esta sección.

6.1.2.4 Obtener los documentos de salida/errores

Un documento de resultados[281] puede contener los URI de archivo o los URI relativos de los documentos de errores[282], documentos de mensajes (como los registros) y documentos de salida[283]. (En algunos casos[284] el documento de resultados no contiene ningún URI). Los diferentes tipos de URI se describen más abajo[285].

Para acceder a estos documentos por HTTP:

1. Amplíe el URI relativo[286] del archivo del documento de resultados hasta su URI absoluto.
2. Use el URI ampliado en una solicitud HTTP[286] `GET`[286] para acceder al archivo.

URI (en el documento de resultados) de los documentos de errores, de mensajes y de salida

El documento de resultados contiene los URI de los documentos de errores, de mensajes y de salida. Los documentos de errores y de mensajes están relacionados con el trabajo y los genera el servidor. Siempre se guardan en el servidor, en el directorio del trabajo. Los documentos de salida (como los resultantes de transformaciones XSLT) se pueden guardar en una de estas dos ubicaciones:

- En cualquier ubicación de archivo a la que pueda acceder el servidor. Para guardar los archivos en cualquier ubicación, debe configurar el servidor para permitir acceso no restringido[259] al cliente (configuración predeterminada).
- En el directorio de trabajo del servidor. El servidor se configura[258] para restringir el acceso al cliente.

Si un cliente especifica que se debe crear un archivo de salida, la ubicación en la que se guarda el archivo depende de la opción `server.unrestricted-filesystem-access`[259] del archivo de configuración del servidor.

- Si el acceso no está restringido, el archivo se guardará en la ubicación especificada por el ciente y el URI devuelto para el documento será un URI de archivo.
- Si el acceso está restringido, el archivo se guardará en el directorio del trabajo y su URI será un URI relativo. Además, se creará una asignación entre este URI relativo y la URL de archivo especificada por el cliente (ver el ejemplo del apartado *Documento de resultados con los URI de los documentos de salida*[283])

Para resumir, hay 3 tipos de URI:

URI de archivo de los documentos de errores / mensajes
Estos documentos se guardan en el servidor en el directorio del trabajo. Los URI de archivo tienen este formato:

```
file:///<dir-raíz-de-salida>/JOBID/mensaje.doc
```

URI de archivo de los documentos de salida
Estos documentos se guardan en cualquier ubicación. Los URI de archivo tienen este formato:

```
file:///<ruta-del-archivo>/salida.doc
```

URI HTTP de los documentos de errores/mensajes/salida
Estos documentos se guardan en el servidor en el directorio del trabajo. Los URI son relativos a la dirección del servidor y deben ampliarse a un URI HTTP. Este es su formato:

```
/vN/results/JOBID/error/error.txt      para documentos de errores
/vN/results/JOBID/output/verbose.log   para documentos de mensajes
/vN/results/JOBID/output/1             para documentos de salida
```

En el caso de los documentos de salida, se dan las asignaciones de salida (*ver fragmento de ejemplo* 283). Estas asignaciones unen cada URI de documento de salida del documento de resultados con el documento correspondiente de la solicitud cliente.

Ampliar el URI relativo

Amplíe el URI relativo del documento de resultados 281 a un URI HTTP absoluto añadiendo la dirección del servidor como prefijo al URI relativo. Por ejemplo, si la dirección del servidor es:

```
http://localhost:8087/
```
(la configuración inicial 258)

y el URI relativo de un archivo de errores del documento de resultados 281 es:

```
/v1/results/20008201-219F-4790-BB59-C091C276FED2/error/error.txt
```

entonces la dirección absoluta ampliada será

```
http://localhost:8087/v1/results/20008201-219F-4790-BB59-C091C276FED2/error/error.txt
```

Para más información consulte los apartados Configurar el servidor 258 y Obtener el documento de resultados 281.

Utilice una solicitud HTTP `GET` para acceder a los archivos

Utilice el URI ampliado en una solicitud HTTP `GET` para obtener el archivo que necesita. RaptorXML Server devuelve el documento solicitado.

6.1.2.5 Liberar espacio tras el procesamiento

RaptorXML Server guarda en el disco duro el documento de resultados, archivos temporales, documentos de errores y documentos de salida relacionados con un trabajo. Hay dos maneras de eliminar estos archivos:

- Suministrando el URI del documento de resultados 282 con un método HTTP `DELETE`. Esto elimina todos los archivos relacionados con el trabajo indicado con el URI del documento de resultados, incluidos los documentos de salida y de errores.
- Eliminando a mano los archivos del servidor (se necesitan derechos de administrador).

© 2017-2023 Altova GmbH

La estructura del URI que debe usar con el método HTTP `DELETE` aparece a continuación. Observe que el URI está formado por la dirección del servidor más el URI relativo del documento de resultados.

Método HTTP	URI
DELETE	http://localhost:8087/v1/result/D405A84A-AB96-482A-96E7-4399885FAB0F

Para encontrar el directorio de salida de un trabajo en el disco, construya el URI de esta forma:

[<server.output-root-dir> *ver archivo de configuración del servidor*[259]] + [IDtrabajo[281]]

Nota: es posible que se generen muchos documentos de salida y de errores, por lo que aconsejamos supervisar el uso del disco duro y programar eliminaciones frecuentes, en función de su entorno y requisitos.

6.1.3 Ejemplo en C# para API REST

La aplicación RaptorXML Server que tiene instalada contiene un proyecto en C# que accede a su interfaz cliente REST para ejecutar trabajos. El proyecto de ejemplo tiene dos partes:

- **`RaptorXMLREST.cs`**: Una clase contenedora en C# que usa el mecanismo REST para comunicarse con RaptorXML Server vía HTTP.
- **`Program.cs`**: El código de programa C# que define los trabajos que se envían a RaptorXML Server con el contenedor REST.

Encontrará la descripción de estas dos partes en los apartados siguientes: Contenedor C# para API REST[288] y Código de programa para solicitudes REST[288].

Recuerde que puede usar cualquier contenedor REST compatible para el código C#. La razón principal por la que creamos nuestro propio contenedor es para que el código de programa C# se pueda integrar de forma más precisa con la clase contenedora, lo que permite entender mejor la interfaz REST de RaptorXML Server.

Ubicación y uso del ejemplo C#

El proyecto de ejemplo se encuentra en la carpeta `C:\Archivos de programa (x86)\Altova\RaptorXML Server2024\examples\REST_API\C#_RaptorREST_API`.

Este proyecto se creó con Visual Studio 2019, por lo que necesitará esta versión o una posterior para generar y ejecutar el proyecto. Recuerde que los archivos de ejemplo en C# están en la carpeta Archivos de programa, por lo que deberá abrir Visual Studio con derechos de administrador para poder acceder a ellos. También puede copiar el proyecto de ejemplo a una ubicación distinta y adaptar el proyecto; en este caso no olvide realizar los ajustes necesarios en el proyecto.

6.1.3.1 Contenedor C# para API REST

La clase contenedora se define en el archivo C# **RaptorXMLREST.cs** y se llama **RaptorXMLRESTAPI**.

Lo que se definen son varias clases clave para enviar y recibir solicitudes HTTP con REST.

- Comando
- MultiPartCommand
- CommandResponse
- ResultDocument

Define estas funciones:

- pollCommandResult
- fetchCommandResult
- sendRequest
- cleanupResults

Para ver cómo implementa el contenedor la API REST lea el apartado Solicitudes cliente[267], donde se explica el funcionamiento de la API REST. A continuación puede leer el código C# de la clase contenedora para ver cómo implementa el contenedor el código C# para la API REST.

Por ejemplo, si quiere ver cómo se envía un comando a RaptorXML Server con código C#, puede hacer lo siguiente:

- La interfaz REST permite enviar comandos a RaptorXML Server con solicitudes HTTP **POST**. Este mecanismo se describe en el apartado Iniciar trabajos con POST[269].
- La cuestión siguiente es: ¿Cómo pasa el contenedor el comando a la API REST? El mecanismo que usa el contenedor se define en su clase **Command**. Abra el archivo **RaptorXMLREST.cs** para ver el código de la clase **Command**.
- Por último, para ver cómo instancia el código de programa[288] la clase **Command** del contenedor, consulte el código de los tres trabajos de ejemplo[288].

6.1.3.2 Código de programa para solicitudes REST

El código de programa C# que contiene los trabajos para RaptorXML Server se define en el archivo C# **Program.cs**. Este código usa las clases que se definieron en el contenedor C# para API REST[288]`para crear las solicitudes REST que se envían a RaptorXML Server.

En el código de programa verá tres casos de uso con los que aprenderá a usar la API REST de RaptorXML Server.

- Validar archivos XML de referencia[289] con el comando **valany**[209] de RaptorXML Server. Se hace referencia al archivo de esquema de dentro del archivo XML y no es necesario proporcionar un argumento al comando.
- Validar dos archivos XML[289] con el comando **valany**[209] de RaptorXML Server. Los dos archivos XML, además del archivo de esquema que se usa para validar, se cargan con el comando como cadenas de texto adjuntas. Una vez completadas las validaciones, los resultados se devuelven juntos.

 © 2017-2023 Altova GmbH

- Cargar y transformar archivos XML con un archivo XSLT[290]. Los dos tipos de archivos se cargan vía REST. El comando de RaptorXML Server que se usa es **xslt**[128]. Una vez completada la transformación, el programa obtiene el documento resultante.

Más abajo puede ver con más detalle el código de estos tres casos.

Subsanar errores

Si hay errores, la función **HandleError** que hay al final del código recupera el mensaje de error de la respuesta del servidor[279].

Caso 1: Validar un archivo XML de referencia (comando simple)

El código de programa de este caso usa clases y funciones del contenedor de la API REST para establecer y llevar a cabo la comunicación HTTP con RaptorXML Server. Esta es la lógica del código:

RaptorXMLRESTAPI.Command	Indica el comando de RaptorXML Server al que se llama, que es **valany**, y el archivo que se usa como argumento de ese comando.
RaptorXMLRESTAPI.CommandResponse	Pone la respuesta del servidor a la solicitud de validación en la variable **jsonResponse**. Tenga en cuenta que los trabajos de validación se devuelven como "OK" o "Fallido"[281].
RaptorXMLRESTAPI.ResultDocument	Obtiene el documento de resultados devuelto por el servidor y, si no hay errores, muestra el resultado de la validación.

Caso 2: Valida dos archivos XML en base a un archivo XSD (comando multiparte).

El código de programa de este caso usa la clase **MultiPartCommand** del contenedor de la API REST para establecer y llevar a cabo la comunicación HTTP con RaptorXML Server. Como queremos cargar los archivos dentro del cuerpo de la solicitud, el encabezado del mensaje debe indicar que el tipo de contenido es multipart/form-data[269]. La clase **MultiPartCommand** del contenedor se usa para establecer la comunicación HTTP REST en concordancia. El código de este caso está estructurado como sigue:

RaptorXMLRESTAPI.MultiPartCommand	Indica el comando RaptorXML Server al que se llama, que es **valany**, y después usa la función **AppendAttachment** de la clase para cargar los dos archivos XML y el archivo de esquema. Los archivos se cargan como cadenas de texto. El servidor devuelve una respuesta con el resultado de la validación de los dos archivos; esta respuesta se almacena en la variable **jsonResponse**.
RaptorXMLRESTAPI.fetchCommandResult	Obtiene el documento de resultados que devuelve el servidor y, si no hay errores, muestra el resultado de la validación.
RaptorXMLRESTAPI.cleanupResults	Esta función del contenedor usa el método DELETE de HTTP para eliminar el documento de resultados, los archivos temporales y los documentos de error y resultados relacionados con el trabajo.

Caso 3: Transformación XSLT de archivos XML y XSLT (comando multiparte)

El código de programa de este caso es parecido al del caso anterior y usa la clase **MultiPartCommand** para realizar una transformación XSLT y mostrar el resultado en una caja de mensaje. Los archivos XML y XSLT de la transformación se cargan con la solicitud. El comando XSLT deRaptorXML Server también admite opciones, así que en este caso puede ver cómo añadir opciones con la interfaz REST (en el ejemplo para ello se usa la función **RaptorXMLRESTAPI.AppendOption**). A continuación comentamos los puntos principales del código.

RaptorXMLRESTAPI.MultiPartCommand	Indica el comando de RaptorXML Server al que se llama, que es **XSLT**, y después usa (i) la función **AppendAttachment** de la clase para cargar los archivos XML y XSLT, y (ii) la función **AppendOption** para obtener opciones para la línea de comandos de comandos de RaptorXML Server. Los archivos cargados se envían como cadenas de texto. El servidor devuelve una respuesta con el resultado de la validación de los dos archivos; esta respuesta se almacena en la variable **jsonResponse**.
RaptorXMLRESTAPI.fetchCommandResult	Obtiene el documento de resultados que devuelve el servidor y, si no hay errores, muestra el resultado de la validación.
RaptorXMLRESTAPI.cleanupResults	Esta función del contenedor usa el método DELETE de HTTP para limpiar el documento de resultados, los archivos temporales y los documentos de error y resultados relacionados con el trabajo.

© 2017-2023 Altova GmbH

6.2 APIs COM y .NET

RaptorXML Server tiene asignada una licencia en el equipo en el que está instalado. La interfaz .NET está construida como un envoltorio en torno a la interfaz COM. Las interfaces COM y .NET de RaptorXML Server utilizan una sola API: la API de COM/.NET API para RaptorXML Server (referencia de objeto 303).

Puede usar RaptorXML Server con:

- Lenguajes de scripting como JavaScript, a través de la interfaz COM
- Lenguajes de programación como C#, a través de la interfaz .NET Framework

6.2.1 Interfaz COM

Durante la instalación, RaptorXML Server se registra automáticamente como objeto de servidor COM. Esto permite invocar a RaptorXML Server desde otras aplicaciones y lenguajes de scripting compatibles con el uso de llamadas COM. Si quiere cambiar la ubicación del paquete de instalación de RaptorXML Server, es mejor desinstalar RaptorXML Server y volver a instalarlo en su nueva ubicación. De este modo el programa de instalación se encarga del proceso de registro.

Comprobar si RaptorXML Server se registró correctamente

Si RaptorXML Server se registró correctamente, el registro incluye las clases `RaptorXML.Server`. Estas clases suelen estar en `HKEY_LOCAL_MACHINE\SOFTWARE\Classes`.

Código de ejemplo

- El ejemplo de código VBScript 291 muestra cómo se puede usar la API de RaptorXML a través de su interfaz COM.
- Para este fragmento de código también ofrecemos un archivo de ejemplo en la carpeta `examples/API` de la carpeta de aplicación de RaptorXML.

6.2.2 Ejemplo de COM: VBScript

Este ejemplo de código VBScript se divide en varias partes:

- Preparación e inicialización del objeto COM de RaptorXML 291
- Validación de un archivo XML 292
- Transformación XSLT y devolución del resultado en forma de cadena de texto 292
- Procesamiento de un documento XQuery y devolución del resultado en forma de archivo 292
- Preparación de la secuencia de ejecución del script y de su punto de entrada 293

```
' El objeto COM de RaptorXML
dim objRaptor

' Inicializar el objeto COM de RaptorXML
sub Init
	objRaptor = Null
	On Error Resume Next
```

```
	' Intentar cargar el objeto COM de 32 bits. No emitir excepciones si no se encuentra
el objeto
	Set objRaptor = WScript.GetObject( "", "RaptorXML.Server" )
	On Error Goto 0
	if ( IsNull( objRaptor ) ) then
		' Intentar cargar el objeto COM de 64 bits (se emite una excepción si no se
encuentra el objeto)
		Set objRaptor = WScript.GetObject( "", "RaptorXML_x64.Server" )
	end if
	' Configurar el servidor: notificación de errores, nombre y puerto del servidor HTTP
(IPv6 localhost en este	ejemplo)
	objRaptor.ErrorLimit = 1
	objRaptor.ReportOptionalWarnings = true
	objRaptor.ServerName = "::1"
	objRaptor.ServerPort = 8087
end sub


' Validar un archivo
sub ValidateXML
	' Obtener una instancia del validador del objeto servidor
	dim objXMLValidator
	Set objXMLValidator = objRaptor.GetXMLValidator()

	' Configurar los datos de entrada
	objXMLValidator.InputFileName = "MiArchivoXML.xml"

	' Validar; en caso de que el archivo no sea válido, notificar el problema devuelto
por RaptorXML
	if ( objXMLValidator.IsValid() ) then
		MsgBox( "La cadena de entrada es válida" )
	else
		MsgBox( objXMLValidator.LastErrorMessage )
	end if
end sub


' Realizar una transformación; devolver el resultado en forma de cadena de texto
sub RunXSLT
	' Obtener una instancia del motor XSLT del objeto servidor
	dim objXSLT
	set objXSLT = objRaptor.GetXSLT

	' Configurar los datos de entrada
	objXSLT.InputXMLFileName = "MiArchivoXML.xml"
	objXSLT.XSLFileName = "MiTransformación.xsl"

	' Ejecutar la transformación; si finaliza correctamente, el resultado se devuelve.
Si no, el motor devuelve  una lista de errores
	MsgBox( objXSLT.ExecuteAndGetResultAsString() )
end sub


' Ejecutar un XQuery; guardar el resultado en un archivo
sub RunXQuery
```

 © 2017-2023 Altova GmbH

```
        ' Obtener una instancia del motor XQuery del objeto servidor
        dim objXQ
        set objXQ = objRaptor.GetXQuery()

        ' Configurar los datos de entrada
        objXQ.InputXMLFileName = "MiArchivoXML.xml"
        objXQ.XQueryFileName = "MiXQuery.xq"

        ' Configurar la serialización (opcional si quiere perfeccionar el formato del
archivo de salida)
        objXQ.OutputEncoding = "UTF8"
        objXQ.OutputIndent = true
        objXQ.OutputMethod = "xml"
        objXQ.OutputOmitXMLDeclaration = false

        ' Ejecutar la consulta; el resultado se serializa en la ruta de acceso indicada
        call objXQ.Execute( "MyQueryResult.xml" )
end sub


' Realizar todas las funciones de muestra
sub main
        Init
        ValidateXML
        RunXSLT
        RunXQuery
end sub

' Punto de entrada del script; ejecutar la función principal
main
```

6.2.3 Interfaz NET

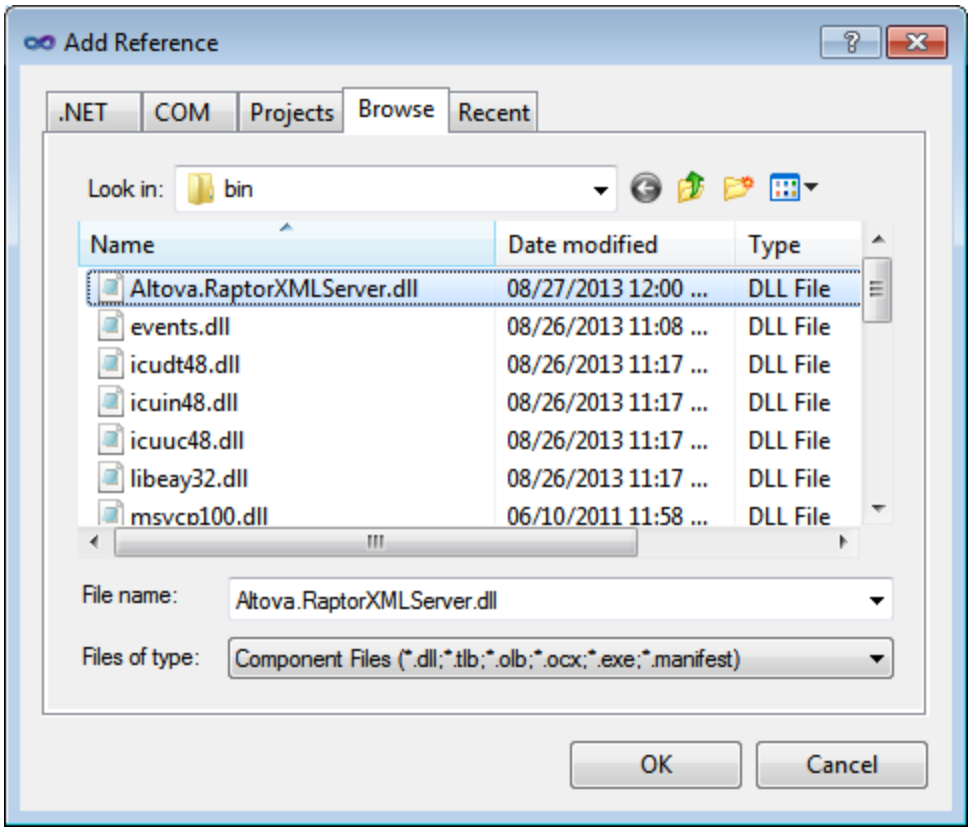
La interfaz .NET está construida como contenedor alrededor de la interfaz COM de RaptorXML. Se ofrece como ensamblado de interoperabilidad principal firmado por Altova y usa el espacio de nombres `Altova.RaptorXMLServer`.

Añadir al proyecto Visual Studio .NET una referencia al DLL de RaptorXML

Para poder usar RaptorXML en su proyecto .NET debe añadir una referencia al DLL de RaptorXML (`Altova.RaptorXMLServer.dll`) en el proyecto. El paquete de instalación de RaptorXML Server contiene un archivo DLL firmado y llamado `Altova.RaptorXMLServer.dll`, que se añadirá automáticamente al caché global de ensamblados cuando se instale RaptorXML usando el programa de instalación de RaptorXML. (Su ubicación suele ser la carpeta `C:\WINDOWS\assembly`.)

Para añadir una referencia a este DLL en un proyecto .NET:

1. Abra el proyecto .NET y haga clic en **Proyecto | Añadir referencia**. Aparece el cuadro de diálogo "Añadir referencia" (*imagen siguiente*).



2. En la pestaña *Browse* vaya a la carpeta: `<RaptorXML application folder>/bin`, seleccione el DLL de RaptorXML `Altova.RaptorXMLServer.dll` y haga clic en **Aceptar**.
3. Seleccione el comando **View | Object Browser** para ver los objetos de la API de RaptorXML.

Cuando `Altova.RaptorXMLServer.dll` esté a disposición de la interfaz .NET y RaptorXML esté registrado como objeto servidor COM, las funciones de RaptorXML estarán disponibles en el proyecto .NET.

Nota: RaptorXML se registra automáticamente como objeto servidor COM durante la instalación. No es necesario registrarlo a mano.

Nota: si recibe un error de acceso, compruebe que tiene los permisos necesarios. Vaya a Servicios de componentes y otorge permisos a la misma cuenta que ejecuta el grupo de aplicaciones que contiene RaptorXML.

Código de ejemplo

El ejemplo de código C# 295 y el de código Visual Basic .NET 297 muestran cómo se puede usar la API de RaptorXML a través de su interfaz .NET. Para estos ejemplos de código también ofrecemos archivos de muestra en la carpeta `examples/serverAPI` de la carpeta de aplicación de RaptorXML.

 © 2017-2023 Altova GmbH

6.2.4 Ejemplo de .NET: C#

Este ejemplo de código C# se divide en varias partes:

- Preparación e inicialización del objeto .NET de RaptorXML 295
- Validación de un archivo XML 295
- Transformación XSLT y devolución del resultado en forma de cadena de texto 296
- Procesamiento de un documento XQuery y devolución del resultado en forma de archivo 296
- Preparar la secuencia de ejecución del código y su punto de entrada 296

```
using System;
using System.Text;
using Altova.RaptorXMLServer;

namespace RaptorXMLRunner
{
    class Program
    {
    // El objeto .NET de RaptorXML
       static ServerClass objRaptorXMLServer;

    // Inicializar el objeto .NET de RaptorXML
       static void Init()
       {
       // Asignar un objeto de RaptorXML
          objRaptorXMLServer = new ServerClass();

       // Configurar el servidor: notificación de errores, nombre y puerto del servidor
HTTP
       // (IPv6 localhost en este ejemplo)
          objRaptorXMLServer.ErrorLimit = 1;
          objRaptorXMLServer.ReportOptionalWarnings = true;
          objRaptorXMLServer.ServerName = "::1"
          objRaptorXMLServer.ServerPort = 8087
       }



       // Validar un archivo
          static void ValidateXML()
          {
          // Obtener una instancia del motor de validación del objeto servidor
             XMLValidator objXMLValidator = objRaptorXMLServer.GetXMLValidator();

          // Configurar los datos de entrada
             objXMLValidator.InputFileName = "MyXMLFile.xml";

          // Validar; si el archivo no es válido,
             notificar el problema devuelto por RaptorXML
             if ( objXMLValidator.IsValid() )
                Console.WriteLine( "La cadena de entrada no es válida" );
```

```
      else
         Console.WriteLine( objXMLValidator.LastErrorMessage );
   }



// Realizar una transformación XSLT y
// devolver el resultado en forma de cadena de texto
   static void RunXSLT()
   {
   // Obtener una instancia del motor XSLT del objeto servidor
      XSLT objXSLT = objRaptorXMLServer.GetXSLT();

      // Configurar los datos de entrada
         objXSLT.InputXMLFileName = "MiArchivoXML.xml";
         objXSLT.XSLFileName = "MiTransformación.xsl";

     // Ejecutar la transformación.
     // Si se ejecuta correctamente, el resultado se devuelve.
     // Si hay errores, se emite una lista de errores
        Console.WriteLine( objXSLT.ExecuteAndGetResultAsString() );
   }



// Ejecutar un XQuery y guardar el resultado en un archivo
   static void RunXQuery()
   {
   // Obtener una instancia del motor XQuery del objeto servidor
      XQuery objXQuery = objRaptorXMLServer.GetXQuery();

     // Configurar los datos de entrada
        objXQuery.InputXMLFileName = exampleFolder + "simple.xml";
        objXQuery.XQueryFileName = exampleFolder + "CopyInput.xq";

    // Configurar la serialización (opcional si quiere mejorar el formato de salida)
       objXQuery.OutputEncoding = "UTF8"
       objXQuery.OutputIndent = true
       objXQuery.OutputMethod = "xml"
       objXQuery.OutputOmitXMLDeclaration = false

    // Ejecutar la consulta. El resultado se serializa en la ruta de acceso indicada
       objXQuery.Execute( "MyQueryResult.xml" );
   }



   static void Main(string[] args)
   {
     try
     {
     // Punto de entrada. Realizar todas las funciones
```

 © 2017-2023 Altova GmbH

```
            Init();
            ValidateXML();
            RunXSLT();
            RunXQuery();
        }
        catch (System.Exception ex)
        {
            Console.WriteLine( ex.Message );
            Console.WriteLine( ex.ToString() );
        }
    }
  }
}
```

6.2.5 Ejemplo de .NET: Visual Basic .NET

Este ejemplo de código Visual Basic se divide en varias partes:

- Preparar e inicializar el objeto .NET de RaptorXML 297
- Validar un archivo XML 297
- Transformación XSLT y devolución del resultado en forma de cadena de texto 298
- Procesamiento de un documento XQuery y devolución del resultado en forma de archivo 298
- Preparación de la secuencia de ejecución del código y de su punto de entrada 298


```
Option Explicit On
Imports Altova.RaptorXMLServer

Module RaptorXMLRunner

   ' El objeto .NET de RaptorXML
     Dim objRaptor As Server

   ' Inicializar el objeto .NET de RaptorXML
     Sub Init()

       ' Asignar un objeto RaptorXML
         objRaptor = New Server()

       ' Configurar el servidor: notificación de errores, nombre y puerto del servidor HTTP
(IPv6 localhost en este ejemplo)
         objRaptor.ErrorLimit = 1
         objRaptor.ReportOptionalWarnings = True
         objRaptor.ServerName = "::1"
         objRaptor.ServerPort = 8087
     End Sub


   ' Validar un archivo
     Sub ValidateXML()

       ' Obtener una instancia del validador del objeto RaptorXML
         Dim objXMLValidator As XMLValidator
```

```
        objXMLValidator = objRaptor.GetXMLValidator()

      ' Configurar los datos de entrada
        objXMLValidator.InputFileName = "MiArchivoXML.xml"

      ' Validar; si el archivo no es válido, notificar el problema devuelto por RaptorXML
        If (objXMLValidator.IsValid()) Then
           Console.WriteLine("La cadena de entrada es válida")
        Else
           Console.WriteLine(objXMLValidator.LastErrorMessage)
        End If
    End Sub


   ' Realizar una transformación; devolver el resultado en forma de cadena de texto
    Sub RunXSLT()

      ' Obtener una instancia del motor XSLT del objeto servidor
        Dim objXSLT As XSLT
        objXSLT = objRaptor.GetXSLT()

      ' Configurar los datos de entrada
        objXSLT.InputXMLFileName = "MiArchivoXML.xml"
        objXSLT.XSLFileName = "MiTransformación.xsl"

      ' Ejecutar la transformación; si se ejecuta correctamente, se devuelve el resultado;
si hay errores, devolver  una lista de errores
        Console.WriteLine(objXSLT.ExecuteAndGetResultAsString())
    End Sub


   ' Ejecutar un XQuery; guardar el resultado en un archivo
    Sub RunXQuery()

      ' Obtener una instancia del motor XQuery del objeto servidor
        Dim objXQ As XQuery
        objXQ = objRaptor.GetXQuery()

      ' Configurar los datos de entrada
        objXQ.InputXMLFileName = "MyXMLFile.xml"
        objXQ.XQueryFileName = "MyQuery.xq"

      ' Configurar la serialización (opcional para mejorar el formato del archivo de
salida)
        objXQ.OutputEncoding = "UTF8"
        objXQ.OutputIndent = true
        objXQ.OutputMethod = "xml"
        objXQ.OutputOmitXMLDeclaration = false

      ' Ejecutar la consulta; el resultado se serializa en la ruta de acceso indicada
        objXQ.Execute( "MyQueryResult.xml" )
    End Sub


    Sub Main()
     ' Punto de entrada; realizar todas las funciones de muestra
```

 © 2017-2023 Altova GmbH

```
        Init()
        ValidateXML()
        RunXSLT()
        RunXQuery()
    End Sub

End Module
```

6.3 API de Java

A la API de RaptorXML también se puede acceder desde código Java. Para ello es necesario que las bibliotecas que aparecen más abajo estén listadas en el parámetro classpath. Esta bibliotecas se instalan en la carpeta `bin` de la carpeta de instalación.

- `RaptorXMLServer.jar`: es la biblioteca que se comunica con el servidor de RaptorXML mediante solicitudes HTTP.
- `RaptorXMLServer_JavaDoc.zip`: es un archivo Javadoc que contiene la documentación de ayuda para la API de Java.

Nota: para poder usar la API de Java, el archivo Jar debe estar en el parámetro classpath. Si quiere, puede copiar el

6.3.1 Resumen de la interfaz Java

La API de Java está incluida en el paquete **`com.altova.raptorxml`**. La clase **`RaptorXML`** ofrece un método de punto de entrada llamado **`getFactory()`** que devuelve objetos `RaptorXMLFactory`[303]. Por tanto se pueden crear instancias `RaptorXMLFactory`[303] con la llamada `RaptorXML.getFactory()`.

La interfaz `RaptorXMLFactory`[303] ofrece métodos para obtener objetos del motor para realizar tareas de validación y procesamiento (como transformaciones XSLT).

RaptorXMLFactory

La interfaz pública de `RaptorXMLFactory`[303] se describe en este fragmento:

```
public interface RaptorXMLFactory
{
   public XMLValidator[320] getXMLValidator();
   public XMLDSig[313] getXQuery();
   public XQuery[332] getXQuery();
   public XSLT[344] getXSLT();
   public void setServerName(String name) throws RaptorXMLException[312];
   public void setServerPath(String path) throws RaptorXMLException[312];
   public void setServerPort(int port) throws RaptorXMLException[312];
   public void setGlobalCatalog(String catalog);
   public void setUserCatalog(String catalog);
   public void setGlobalResourcesFile(String file);
   public void setGlobalResourceConfig(String config);
   public void setErrorFormat(RaptorXMLException[312] format);
   public void setErrorLimit(int limit);
   public void setReportOptionalWarnings(boolean report);
}
```

Para más información consulte la descripción de `RaptorXMLFactory`[303] y de los métodos Java correspondientes. También puede consultar un ejemplo de proyecto Java[301].

© 2017-2023 Altova GmbH

6.3.2 Ejemplo de proyecto Java

El fragmento de código Java que aparece más adelante muestra cómo acceder a las funciones básicas. El código tiene varias partes:

- Busca la carpeta de ejemplos y crea una instancia de objeto COM de RaptorXML 301
- Valida un archivo XML 301
- Realiza una transformación XSLT y devuelve el resultado en forma de cadena de texto 301
- Procesa un documento XQuery y devuelve el resultado en forma de cadena de texto 302
- Ejecuta el proyecto 302

Estas funciones básicas se incluyen en los archivos de la carpeta `examples/API` de la carpeta de aplicación de RaptorXML Server.

```
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();
        }

    }

}
```


© 2017-2023 Altova GmbH

6.4 Referencia sobre las APIs de servidor

En este apartado se describe la API de RaptorXML Server, su modelo de objetos y los detalles de sus interfaces y enumeraciones. La descripción de la API incluye las interfaces de COM y .NET. Aunque la estructura de la API es la misma para las dos interfaces, los nombres de los métodos y las propiedades son diferentes. Por eso cada método, propiedad y enumeración se describen por separado para COM y Java.

El punto de partida para usar las funciones de RaptorXML Server es la interfaz `IServer` 303 (COM/.NET) o la clase `RaptorXMLFactory` 303 (Java).

6.4.1 Interfaces/Clases

El punto de partida para usar las funciones de RaptorXML es la interfaz `IServer` 303 (COM/.NET) o la clase `RaptorXMLFactory` 303 (Java). Este objeto contiene los objetos que proporcionan las funciones de RaptorXML: validación de XML, procesamiento de documentos XQuery y XML Signature, y transformaciones XSLT.

A continuación mostramos la jerarquía del modelo del objeto; las interfaces se describen en detalle en los apartados correspondientes. Los métodos y las propiedades de cada interfaz se describen en el apartado de la interfaz correspondiente.

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

Use la interfaz **IServer/RaptorXMLFactory** para acceder al motor RaptorXML que prefiera. Tenga en cuenta que el nombre de la interfaz en la API COM/.NET es distinto al de la API de Java:

- En COM/.NET: **IServer**
- En Java: **RaptorXMLFactory**

En esta sección se describen los métodos y las propiedades de **IServer/RaptorXMLFactory**.

Método del punto de entrada a la API de Java

La API de Java está incluida en el paquete **com.altova.raptorxml**. La clase **RaptorXML** ofrece un método de punto de entrada llamado **getFactory()** que devuelve objetos `RaptorXMLFactory` 303. Por tanto se pueden crear instancias `RaptorXMLFactory` 303 con la llamada `RaptorXML.getFactory()`.

6.4.1.1.1 Métodos

Los métodos de las interfaces **IServer** (COM/.NET) y **RaptorXMLFactory** (Java) devuelven respectivamente una instancia del motor o la clase de RaptorXML: XMLDSig, XML Validator, XSLT y XQuery.

COM/.NET	Java
GetXMLDsig 304 *(para firmas XML)*	getXMLDsig 304 *(para firmas XML)*
GetXMLValidator 304	getXMLValidator 304
GetXQuery 305	getXQuery 305
GetXSLT 305	getXSLT 305

6.4.1.1.1.1 GetXMLDsig (para firmas XML)

Devuelve una instancia de la interfaz/clase XML (**XMLDSig** 313).

COM y .NET

Firma: **IXMLDSig** 313 **GetXMLDSig()**

Java

Firma: **public XMLDSig** 313 **getXMLDSig()**

6.4.1.1.1.2 GetXMLValidator

Devuelve una instancia del motor de validación XML.

COM y .NET

Firma: **IXMLValidator** 320 **GetXMLValidator()**

Java

Firma: **public XMLValidator** 320 **getXMLValidator()**

 © 2017-2023 Altova GmbH

6.4.1.1.1.3 GetXQuery

Devuelve una instancia del motor XQuery.

COM y .NET

Firma: **IXQuery**[332] **GetXQuery()**

Java

Firma: **public XQuery**[332] **getXQuery()**

6.4.1.1.1.4 GetXSLT

Devuelve una instancia del motor XSLT.

COM y .NET

Firma: **IXSLT**[344] **GetXSLT()**

Java

Firma: **public XSLT**[344] **getXSLT()**

6.4.1.1.2 Propiedades

En esta sección se describen las propiedades de las interfaces **IServer** (COM/.NET) y **RaptorXMLFactory** (Java).

COM/.NET	Java
APIMajorVersion[306]	getAPIMajorVersion[306]
APIMinorVersion[306]	getAPIMinorVersion[306]
APIServicePackVersion[307]	getAPIServicePackVersion[307]
ErrorFormat[307]	setErrorFormat[307]
ErrorLimit[307]	setErrorLimit[307]
GlobalCatalog[308]	setGlobalCatalog[308]

GlobalResourceConfig 308	setGlobalResourceConfig 308
GlobalResourcesFile 308	setGlobalResourcesFile 308
Is64Bit 309	ss64Bit 309
MajorVersion 309	getMajorVersion 309
MinorVersion 309	getMinorVersion 309
ProductName 310	getProductName 310
ProductNameAndVersion 310	getProductNameAndVersion 310
ReportOptionalWarnings 310	setReportOptionalWarnings 310
ServerName 311	setServerName 311
ServerPath 311	setServerPath 311
ServerPort 311	setServerPort 311
ServicePackVersion 312	getServicePackVersion 312
UserCatalog 312	setUserCatalog 312

6.4.1.1.2.1 APIMajorVersion

Devuelve la versión principal de la API en forma de entero. La versión principal de la API puede diferir de la versión principal del producto 309 si la API está conectada a otro servidor.

COM y .NET

Firma: `int APIMajorVersion()`

Java

Firma: `public int getAPIMajorVersion()`

6.4.1.1.2.2 APIMinorVersion

Devuelve la versión secundaria de la API en forma de entero. La versión secundaria de la API puede diferir de la versión secundaria del producto 309 si la API está conectada a otro servidor.

COM y .NET

Firma: `int APIMinorVersion()`

© 2017-2023 Altova GmbH

Java

Firma: `public int getAPIMinorVersion()`

6.4.1.1.2.3 APIServicePackVersion

Devuelve la versión de service pack de la API en forma de entero. La versión de service pack de la API puede diferir de la versión de service pack del producto[312] si la API está conectada a otro servidor.

COM y .NET

Firma: `int APIServicePackVersion()`

Java

Firma: `public int getAPIServicePackVersion()`

6.4.1.1.2.4 ErrorFormat

Establece el formato de error de RaptorXML en uno de los literales de `ENUMErrorFormat` (Text, ShortXML, LongXML).

COM y .NET

Firma: `ErrorFormat(ENUMErrorFormat[356] format)`

Java

Firma: `public void setErrorFormat(ENUMErrorFormat[356] format)`

6.4.1.1.2.5 ErrorLimit

Establece el límite error de validación de RaptorXML. El parámetro `limit` es de tipo `int` (Java), `uint` (COM/.NET). Especifica el número de errores que se deben notificar antes de detener la ejecución. Use el valor `-1` para que `limit` sea ilimitado (es decir, que se notifiquen todos los errores). El valor predeterminado es `100`.

COM y .NET

Firma: `ErrorLimit(uint limit)`

Java

Firma: `public int setErrorLimit(int limit)`

6.4.1.1.2.6 GlobalCatalog

Establece la ubicación, en forma de URL, del archivo de catálogo principal (de punto de entrada). La cadena dada debe ser una URL absoluta que indique la ubicación exacta del archivo de catálogo principal que se debe usar.

COM y .NET

Firma: `GlobalCatalog(string catalog)`

Java

Firma: `public void setGlobalCatalog(string catalog)`

6.4.1.1.2.7 GlobalResourceConfig

Establece la configuración activa del recurso global. El parámetro `config` es de tipo `String` y especifica el nombre de la configuración utilizada por el recurso global activo.

COM y .NET

Firma: `GlobalResourceConfig(string config)`

Java

Firma: `public void setGlobalResourceConfig(string config)`

6.4.1.1.2.8 GlobalResourcesFile

Establece la ubicación, en forma de URL, del archivo XML de recursos globales. La cadena dada debe ser una URL absoluta que indique la ubicación exacta del archivo XML de recursos globales.

COM y .NET

Firma: `GlobalResourcesFile(string url)`

 © 2017-2023 Altova GmbH

Java

Firma: `public void setGlobalResourcesFile(string url)`

6.4.1.1.2.9 Is64Bit

Comprueba si la aplicación es un archivo ejecutable de 64 bits. Devuelve el valor `true` si se trata de una aplicación de 64 bits y el valor `false` si no lo es. *Ejemplo:* para `Altova RaptorXML Server 2024r2sp1(x64)`, devolverá `true`. Si se produce un error, se emite una excepción **RaptorXMLException** 312.

COM y .NET

Firma: `boolean Is64Bit()`

Java

Firma: `public boolean is64Bit()`

6.4.1.1.2.10 MajorVersion

Devuelve la versión principal del producto en forma de entero. *Ejemplo:* para `Altova RaptorXML Server 2014r2sp1(x64)`, devolvería `16` (la diferencia entre la versión principal (`2014`) y el año inicial `1998`). Si se produce un error, se emite la excepción **RaptorXMLException** 312.

COM y .NET

Firma: `int MajorVersion()`

Java

Firma: `public int getMajorVersion()`

6.4.1.1.2.11 MinorVersion

Devuelve la versión secundaria del producto en forma de entero. *Ejemplo:* para `Altova RaptorXML Server 2024r2sp1(x64)`, devolvería `2` (por el número de versión secundaria `r2`). Si se produce un error, se emite la excepción **RaptorXMLException** 312.

COM y .NET

Firma: `int MinorVersion()`

Java

Firma: `public int getMinorVersion()`

6.4.1.1.2.12 ProductName

Devuelve el nombre del producto en forma de cadena. *Ejemplo:* para `Altova RaptorXML Server 2024r2sp1(x64)`, devolvería `Altova RaptorXML Server`. Si se produce un error, se emite la excepción **RaptorXMLException** 312.

COM y .NET

Firma: `string ProductName()`

Java

Firma: `public string getProductName()`

6.4.1.1.2.13 ProductNameAndVersion

Devuelve la versión de service pack del producto en forma de entero. *Ejemplo:* para `Altova RaptorXML Server 2024r2sp1(x64)`, devolvería `Altova RaptorXML Server 2024r2sp1(x64)`. Si se produce un error, se emite la excepción **RaptorXMLException** 312.

COM y .NET

Firma: `string ProductNameAndVersion()`

Java

Firma: `public string getProductNameAndVersion()`

6.4.1.1.2.14 ReportOptionalWarnings

Habilita o deshabilita la notificación de advertencias. El valor `true` habilita las advertencias y `false` las deshabilita.

 © 2017-2023 Altova GmbH

COM y .NET

Firma: `ReportOptionalWarnings(boolean report)`

Java

Firma: `public void setReportOptionalWarnings(boolean report)`

6.4.1.1.2.15 ServerName

Establece el nombre del servidor HTTP. Si se produce un error, se emite una excepción `RaptorXMLException` 312. El parámetro de entrada es una cadena que indica el nombre del servidor HTTP.

COM y .NET

Firma: `ServerName(string name)`

Java

Firma: `public void setServerName(string name)`

6.4.1.1.2.16 ServerPath

Specifies, in the form of a URL, the path to the HTTP server.

COM y .NET

Firma: `ServerPath(string path)`

Java

Firma: `public void setServerPath(string path)`

6.4.1.1.2.17 ServerPort

Establece el puerto del servidor HTTP por el que se accede al servicio. El puerto debe ser fijo y conocido para que las solicitudes HTTP se puedan dirigir correctamente al servicio. Si se produce un error, se emite una excepción `RaptorXMLException` 312. El parámetro de entrada es un entero que especifica el puerto de acceso del servidor HTTP.

COM y .NET

Firma: `ServerPort(int port)`

Java

Firma: `public void setServerPort(int port)`

6.4.1.1.2.18 ServicePackVersion

Devuelve la versión de service pack del producto como entero. *Ejemplo*: para `RaptorXML Server 2024r2sp1(x64)`, devolvería `1` (por el número de versión de service pack `sp1`). Si se produce un error, se emite una excepción **RaptorXMLException** 312.

COM y .NET

Firma: `int ServicePackVersion()`

Java

Firma: `public int getServicePackVersion()`

6.4.1.1.2.19 UserCatalog

Establece la ubicación, en forma de URL, del archivo de catálogo del usuario personal. La cadena dada debe ser una URL absoluta que indique la ubicación exacta del archivo de catálogo personal que se debe usar.

COM y .NET

Firma: `UserCatalog(string userCatalog)`

Java

Firma: `public void setUserCatalog(string userCatalog)`

6.4.1.2 RaptorXMLException

Genera una excepción que contiene información sobre un error ocurrido durante el procesamiento. El parámetro `message` ofrece información sobre el error.

 © 2017-2023 Altova GmbH

COM y .NET

Firma: `RaptorXMLException(string message)`

Java

Firma: `public void RaptorXMLException(string message)`

6.4.1.3 XMLDSig (para firmas XML)

Los métodos de la interfaz/clase **`IXMLDSig/XMLDSig`** se puede usar para firmar documentos XML, verificar documentos firmados, actualizar con una nueva firma documentos firmados que se han modificado y eliminar firmas.

Tenga en cuenta que el nombre de la interfaz en la API COM/.NET es distinto que el de la clase en la API de Java:

- En COM/.NET: **`IXMLDSig`**
- En Java: **`XMLDSig`**

6.4.1.3.1 Métodos

En esta sección se describen los métodos de la interfaz **`IXMLDSig`** (COM/.NET) y la clase **`XMLDSig`** (Java).

6.4.1.3.1.1 ExecuteRemove

Elimina la firma XML del archivo XML firmado y guarda el documento no firmado en una ubicación de salida definida por `outputPath`, una cadena que contiene la URL de la ubicación del archivo. Si esta operación se puede realizar sin problemas, el resultado es `true`; en caso contrario el resultado es `false`.

COM y .NET

Firma: `boolean ExecuteRemove(string outputPath)`

Java

Firma: `public boolean executeRemove(string outputPath)`

6.4.1.3.1.2 ExecuteSign

Firma el documento XML conforme a las opciones de firma especificadas (dadas en los parámetros `signatureType` y `canonicalizationMethod`; *véase xmlsignature-sign CLI command* 196 *para consultar los valores disponibles*). El archivo de salida lo define `outputPath`, una cadena que contiene la URL del archivo de salida. Si esta operación se realiza con éxito, el resultado es `true`; en caso contrario el resultado es `false`.

COM y .NET

Firma: `boolean ExecuteSign(string outputPath, string signatureType,string canonicalizationMethod)`

Java

Firma: `public boolean executeSign(string outputPath, string signatureType,string canonicalizationMethod)`

6.4.1.3.1.3 ExecuteUpdate

Actualiza la firma XML en el documento XML firmado. Si el documento se ha modificado, la firma XML actualizada será distinta; de lo contrario, la firma actualizada será la misma que al principio. El archivo de salida lo especifica `outputPath`, una cadena que contiene la URL del archivo con la firma actualizada. Si esta operación se realiza con éxito, el resultado es `true`; en caso contrario el resultado es `false`.

Es necesario especificar una de estas propiedades: (i) la propiedad HMAC secret key 318 o (ii) las propiedades certificate-name 316 y certificate-store 316. Si hay opciones de certificación especificadas, estas deben coincidir con las que se usaron previamente para firmar el documento XML. (Tenga en cuenta que las opciones de almacén de certificados de momento no son compatibles con Linux ni con macOS.)

COM y .NET

Firma: `boolean ExecuteUpdate(string outputPath)`

Java

Firma: `public boolean executeUpdate(string outputPath)`

6.4.1.3.1.4 ExecuteVerify

Devuelve el resultado de la verificación de la firma: `true` si la verificación es correcta, `false` en caso contrario.

COM y .NET

© 2017-2023 Altova GmbH

Firma: `boolean ExecuteVerify()`

Java

Firma: `public boolean executeVerify()`

6.4.1.3.2 Propiedades

En esta sección se describen las propiedades de la interfaz **`IXMLDSig`** (COM/.NET) y de la clase **`IXMLDSig`** (Java).

6.4.1.3.2.1 AbsoluteReferenceUri

Indica si la URI del documento firmado se debe leer como absoluta (`true`) o como relativa (`false`). El valor predeterminado es `false`.

COM y .NET

Firma: `AbsoluteReferenceUri(boolean absoluteuri)`

Java

Firma: `public void setAbsoluteReferenceUri(boolean absoluteuri)`

6.4.1.3.2.2 AppendKeyInfo

Indica si se debe incluir el elemento `KeyInfo` en la firma o no. El valor predeterminado es `false`.

COM y .NET

Firma: `AppendKeyInfo(boolean include)`

Java

Firma: `public void setAppendKeyInfo(boolean inlude)`

6.4.1.3.2.3 CertificateName

El nombre del certificado usado para firmar.

Windows
Es el nombre Subject de un certificado del **--certificate-store** (almacén de certificados) seleccionado.

Ejemplo de lista de certificados (con PowerShell)

```
% ls cert://CurrentUser/My
PSParentPath: Microsoft.PowerShell.Security\Certificate::CurrentUser\My
Thumbprint Subject
---------- -------
C9DF64BB0AAF5FA73474D78B7CCFFC37C95BFC6C CN=certificate1
... CN=...
```

Ejemplo: **--certificate-name==certificate1**

Linux/MacOS
--certname indica el nombre de archivo de un certificado X.509v3 codificado mediante PEM y con clave privada. Este tipo de archivos normalmente tiene la extensión `.pem`.

Ejemplo: **--certificate-name==/path/to/certificate1.pem**

COM y .NET

Firma: **CertificateName(string name)**

Java

Firma: **public** void **setCertificateName(string name)**

6.4.1.3.2.4 CertificateStore

La ubicación en la que se almacena el certificado especificado con **--certificate-name**.

Windows
El nombre de un certificado almacenado bajo **cert://CurrentUser**. Los almacenes de certificados disponibles se pueden mostrar (con PowerShell) usando **% ls cert://CurrentUser/**. Los certificados se mostrarían así:

```
Name : TrustedPublisher
Name : ClientAuthIssuer
Name : Root
Name : UserDS
```

 © 2017-2023 Altova GmbH

```
Name : CA
Name : ACRS
Name : REQUEST
Name : AuthRoot
Name : MSIEHistoryJournal
Name : TrustedPeople
Name : MyCertStore
Name : Local NonRemovable Certificates
Name : SmartCardRoot
Name : Trust
Name : Disallowed
```

Ejemplo: **`--certificate-store==MyCertStore`**

Linux/MacOS
La opción **`--certstore`** de momento no es compatible.

COM y .NET

Firma: **`CertificateStore(string filelocation)`**

Java

Firma: **`public void setCertificateStore(string filelocation)`**

6.4.1.3.2.5 DigestMethod

El algoritmo que se usa para computar el valor del resumen (digest) dentro del archivo XML de entrada. Los valores disponibles son: `sha1|sha256|sha384|sha512`.

COM y .NET

Firma: **`DigestMethod(string algo)`**

Java

Firma: **`public void setDigestMethod(string algo)`**

6.4.1.3.2.6 HMACOutputLength

Trunca la salida del algoritmo HMAC a los bits de `length`. Si así se especifica, este valor es

- un múltiplo de 8
- mayor que 80

- más largo que la mitad de la longitud del resultado del algoritmo hash subyacente.

COM y .NET

Firma: `HMACOutputLength(int length)`

Java

Firma: `public void setHMACOutputLength(int length)`

6.4.1.3.2.7 HMACSecretKey

La clave secreta compartida HMAC; debe tener una longitud mínima de seis caracteres.

COM y .NET

Firma: `HMACSecretKey(string key)`

Java

Firma: `public void setHMACSecretKey(string key)`

6.4.1.3.2.8 InputXMLFileName

Establece la ubicación, en forma de URL, del documento XML que se debe procesar. La cadena dada debe ser una URL absoluta que indique la ubicación exacta del archivo XML.

COM y .NET

Firma: `InputXMLFileName(string filepath)`

Java

Firma: `public void setInputXMLFileName(string filepath)`

 © 2017-2023 Altova GmbH

6.4.1.3.2.9 LastErrorMessage

Recupera una cadena que es el último mensaje de error del motor de RaptorXML.

COM y .NET

Firma: `string LastErrorMessage()`

Java

Firma: `public string getLastErrorMessage()`

6.4.1.3.2.10 SignatureMethod

Indica el algoritmo que se debe usar para generar la firma.
Cuando se usa un certificado
Si se especifica un certificado, el `SignatureMethod` es opcional y el valor que se asigna a este parámetro deriva del certificado. Si así se especifica, debe coincidir con el algoritmo usado en el certificado. Ejemplo: `rsa-sha256`.

Cuando se usa una clave secreta hmac
Cuando se usa una clave secreta HMAC, `SignatureMethod` es obligatorio. El valor debe ser compatible con HMAC:

- `hmac-sha256`
- `hmac-sha386`
- `hmac-sha512`
- `hmac-sha1` (no recomendado por la especificación)

Ejemplo: **`hmac-sha256`**
.

COM y .NET

Firma: `SignatureMethod(string algo)`

Java

Firma: `public void setSignatureMethod(string algo)`

6.4.1.3.2.11 Transforms

Especifica las transformaciones de firma XML aplicadas al documento de entrada. Los valores compatibles son:

- `REC-xml-c14n-20010315` para Canonical XML 1.0 (sin comentarios)
- `xml-c14n11` para Canonical XML 1.1 (sin comentarios)
- `xml-exc-c14n#` para Exclusive XML Canonicalization 1.0 (sin comentarios)
- `REC-xml-c14n-20010315#WithComments` para Canonical XML 1.0 (con comentarios)
- `xml-c14n11#WithComments` para Canonical XML 1.1 (con comentarios)
- `xml-exc-c14n#WithComments` para Exclusive XML Canonicalization 1.0 (con comentarios)
- `base64`
- Extensión de Altova `strip-whitespaces`

COM y .NET

Firma: `Transforms(string value)`

Java

Firma: `public void setTransforms(string value)`

6.4.1.3.2.12 WriteDefaultAttributes

Determina si se deben incluir valores de atributo predeterminados de la DTD en el documento firmado.

COM y .NET

Firma: `WriteDefaultAttributes(boolean write)`

Java

Firma: `public void setWriteDefaultAttributes(boolean write)`

6.4.1.4 XMLValidator

La interfaz/clase **`IXMLValidator/XMLValidator`** ofrece métodos para (i) validar varios tipos de documentos, (ii) comprobar si el formato de los documentos es correcto y (iii) extraer un esquema Avro de un binario Avro. También puede añadir procesamiento mediante un script en Python.

Tenga en cuenta que el nombre de la interfaz en la API COM/.NET es distinto que el de la clase en la API de Java:

- En COM/.NET: **`IXMLValidator`**
- En Java: **`XMLValidator`**

 © 2017-2023 Altova GmbH

6.4.1.4.1 Métodos

En esta sección se describen los métodos de la interfaz **IXMLValidator** (COM/.NET) y de la clase **XMLValidator** (Java).

6.4.1.4.1.1 AddPythonScriptFile

Establece la ubicación del archivo de script Python. La cadena dada debe ser una URL absoluta que indique la ubicación exacta del archivo Python. El script de Python se procesará con un paquete Python combinado con RaptorXML Server. La versión de ese paquete Python es 3.11.5.

COM y .NET

Firma: **AddPythonScriptFile(**string **filepath)**

Java

Firma: **public** void **addPythonScriptFile(**string **filepath)**

6.4.1.4.1.2 ClearPythonScriptFile

Borra los archivos de comando de Python añadidos con el método **AddPythonScriptFile** o con la propiedad **PythonScriptFile**.

COM y .NET

Firma: **ClearPythonScriptFile()**

Java

Firma: **public** void **clearPythonScriptFile()**

6.4.1.4.1.3 ExtractAvroSchema

Extrae un esquema Avro de un archivo binario. El parámetro `outputPath` es una URL absoluta que especifica la ubicación de salida. El resultado es `true` si la extracción finaliza correctamente. De lo contrario, el resultado es `false`. Si se produce un error, se emite una excepción **RaptorXMLException** 312. Use `getLastErrorMessage` para obtener más información.

COM y .NET

Firma: `ExtractAvroSchema(string outputPath)`

Java

Firma: `public void extractAvroSchema(string outputPath)`

6.4.1.4.1.4 IsValid

Devuelve el resultado de la validación del documento XML, documento de esquema o documento DTD. El tipo de documento que se debe validar viene dado por el parámetro `type`, que toma como valor un literal de la enumeración ENUMValidationType[360]. Si el documento es válido, el resultado es `true`. Si no lo es, el resultado es `false`. Si se produce un error, se emite una excepción **RaptorXMLException**[312]. Use `getLastErrorMessage` para obtener más información.

COM y .NET

Firma: `boolean IsValid(ENUMValidationType[360] type)`

Java

Firma: `public boolean isValid(ENUMValidationType[360] type)`

6.4.1.4.1.5 IsWellFormed

Devuelve el resultado de la comprobación de formato del documento XML o DTD. El tipo de documento que se debe revisar se especifica con el parámetro `type`, que toma como valor un literal de la enumeración ENUMWellformedCheckType[361]. Si el documento tiene el formato correcto, el resultado es `true`. De lo contrario devuelve `false`. Si se produce un error, se emite la excepción **RaptorXMLException**[312]. Use `getLastErrorMessage` para obtener más información.

COM y .NET

Firma: `boolean isWellFormed(ENUMWellformedCheckType[360] type)`

Java

Firma: `public boolean isWellFormed(ENUMWellformedCheckType[360] type)`

 © 2017-2023 Altova GmbH

6.4.1.4.2 Propiedades

En esta sección se describen las propiedades de la interfaz **IXMLValidator** (COM/.NET) y de la clase **XMLValidator** (Java).

6.4.1.4.2.1 AssessmentMode

Establece el modo de evaluación de la validación XML (`Strict/Lax`), que viene dado por un literal de ENUMAssessmentMode[355].

COM y .NET

Firma: **AssessmentMode(**ENUMAssessmentMode[355] **mode)**

Java

Firma: **public** void **setAssessmentMode(**ENUMAssessmentMode[355] **mode)**

6.4.1.4.2.2 AvroSchemaFileName

Establece la ubicación, en forma de URL, del esquema Avro externo que se debe usar. La cadena dada debe ser una URL absoluta que indique la ubicación exacta del archivo de esquema Avro.

COM y .NET

Firma: **AvroSchemaFileName(**string **url)**

Java

Firma: **public** void **setAvroSchemaFileName(**string **url)**

6.4.1.4.2.3 AvroSchemaFromText

Aporta el contenido del documento de esquema Avro que se debe usar. La cadena dada es el contenido del documento de esquema Avro que se debe usar.

COM y .NET

Firma: **AvroSchemaFromText(**string **avroschema)**

Java

Firma: `public void setAvroSchemaFromText(string avroschema)`

6.4.1.4.2.4 DTDFileName

Establece la ubicación, en forma de URL, del documento DTD que se debe usar para la validación. La cadena dada debe ser una URL absoluta que indique la ubicación exacta del documento DTD.

COM y .NET

Firma: `DTDFileName(string url)`

Java

Firma: `public void setDTDFileName(string url)`

6.4.1.4.2.5 DTDFromText

Aporta el contenido del documento DTD que se debe usar para la validación. La cadena dada es el contenido del documento DTD que se debe usar.

COM y .NET

Firma: `DTDFromText(string dtdtext)`

Java

Firma: `public void setDTDFromText(string dtdtext)`

6.4.1.4.2.6 EnableNamespaces

Habilita el procesamiento preparado para espacios de nombres. Esto ayuda en la revisión de instancias XML para buscar errores debidos a espacios de nombres incorrectos. El valor `true` habilita un procesamiento preparado para espacios de nombres. El valor `false` lo deshabilita. El valor predeterminado es `false`.

COM y .NET

Firma: `EnableNamespaces(boolean enableNS)`

 © 2017-2023 Altova GmbH

Java

Firma: `public void setEnableNamespaces(boolean enableNS)`

6.4.1.4.2.7 InputFileArray

Aporta una matriz de direcciones URL de archivos XML que se deben usar como datos de entrada. La propiedad aporta un objeto que contiene, como cadenas, las URL absolutas de cada archivo XML.

COM y .NET

Firma: `InputFileArray(object fileArray)`

Java

Firma: `public void setInputFileArray(object fileArray)`

6.4.1.4.2.8 InputFileName

Establece la ubicación, en forma de URL, de los datos XML de entrada que se deben procesar. La cadena dada debe ser una URL absoluta que indique la ubicación exacta del archivo de entrada.

COM y .NET

Firma: `InputFileName(string filepath)`

Java

Firma: `public void setInputFileName(string filepath)`

6.4.1.4.2.9 InputFromText

Aporta el contenido del documento XML que se debe procesar. La cadena dada es el contenido del documento XML que se debe procesar.

COM y .NET

Firma: `InputFromText(string doc)`

Java

Firma: `public void setInputFromText(string doc)`

6.4.1.4.2.10 InputTextArray

Aporta una matriz de direcciones URL de los archivos de texto que se deben usar como datos de entrada. La propiedad aporta un objeto que contiene, como cadenas, las direcciones URL absolutas de cada uno de los archivos de texto.

COM y .NET

Firma: `InputTextArray(object textfileArray)`

Java

Firma: `public void setInputTextArray(object textfileArray)`

6.4.1.4.2.11 InputXMLFileName

Establece la ubicación, en forma de URL, del documento XML que se debe procesar. La cadena dada debe ser una URL absoluta que indique la ubicación exacta del archivo XML.

COM y .NET

Firma: `InputXMLFileName(string url)`

Java

Firma: `public void setInputXMLFileName(string url)`

6.4.1.4.2.12 InputXMLFromText

Aporta el contenido del documento XML que se debe procesar. La cadena dada es el contenido del documento XML que se debe procesar.

COM y .NET

Firma: `InputXMLFromText(string xml)`

 © 2017-2023 Altova GmbH

Java

Firma: `public void setInputXMLFromText(string xml)`

6.4.1.4.2.13 Json5

Si se establece en `true` se habilita la compatibilidad con JSON 5.

COM y .NET

Firma: `Json5(boolean json5)`

Java

Firma: `public void setJson5(boolean json5)`

6.4.1.4.2.14 JSONSchemaFileName

Establece como URL la ubicación del archivo de esquema JSON que se usará para la validación de documentos de instancia. La cadena dada debe ser una URL absoluta que indique la ubicación exacta del archivo de esquema JSON.

COM y .NET

Firma: `JSONSchemaFileName(string url)`

Java

Firma: `public void setJSONSchemaFileName(string url)`

6.4.1.4.2.15 JSONSchemaFromText

Da una cadena que consiste en el contenido de texto del documento de esquema JSON que se usará para validar el documento de instancia JSON.

COM y .NET

Firma: `JSONSchemaFromText(string jsonschema)`

Java

Firma: `public void setJSONSchemaFromText(string jsonschema)`

6.4.1.4.2.16 LastErrorMessage

Recupera una cadena que es el último mensaje de error del motor de RaptorXML.

COM y .NET

Firma: `string LastErrorMessage()`

Java

Firma: `public string getLastErrorMessage()`

6.4.1.4.2.17 ParallelAssessment

Habilita o deshabilita la evaluación de la validez de esquemas en paralelo 238.

COM y .NET

Firma: `ParallelAssessment(boolean enable)`

Java

Firma: `public void setParallelAssessment(boolean enable)`

6.4.1.4.2.18 PythonScriptFile

Establece la ubicación del archivo de script Python. La cadena dada debe ser una URL absoluta que indique la ubicación exacta del archivo Python. El script de Python se procesará con un paquete Python combinado con RaptorXML Server. La versión de ese paquete Python es 3.11.5.

COM y .NET

Firma: `PythonScriptFile(string filepath)`

 © 2017-2023 Altova GmbH

Java

Firma: `public void setPythonScriptFile(string filepath)`

6.4.1.4.2.19 SchemaFileArray

Aporta la colección de archivos de esquema XML que se usarán como esquemas XML externos. Los archivos se identifican por medio de su dirección URL. La entrada es una colección de cadenas, cada una de las cuales es la URL absoluta de un archivo de esquema XML.

COM y .NET

Firma: `SchemaFileArray(object urlArray)`

Java

Firma: `public void setSchemaFileArray(object urlArray)`

6.4.1.4.2.20 SchemaFileName

Establece la ubicación, en forma de URL, del documento de esquema XML que se debe validar. La cadena dada debe ser una URL absoluta que indique la ubicación exacta del archivo de esquema XML.

COM y .NET

Firma: `SchemaFileName(string filepath)`

Java

Firma: `public void setSchemaFileName(string filepath)`

6.4.1.4.2.21 SchemaFromText

Aporta el contenido del documento de esquema XML que se debe usar. La cadena dada es el contenido del documento de esquema XML que se debe usar.

COM y .NET

Firma: `SchemaFileName(string xsdText)`

Java

Firma: `public void setSchemaFileName(string xsdText)`

6.4.1.4.2.22 SchemaImports

Especifica cómo se deben realizar las importaciones de esquema, en función de los valores de atributo de los elementos `xs:import`. El tipo de importación se especifica con el literal de `ENUMSchemaImports` que se aporta.

COM y .NET

Firma: `SchemaImports(ENUMSchemaImports 358 importOption)`

Java

Firma: `public void setSchemaImports(ENUMSchemaImports 358 importOption)`

6.4.1.4.2.23 SchemalocatiionHints

Especifica el mecanismo que se debe utilizar para encontrar el esquema. El mecanismo se especifica por medio del literal de `ENUMLoadSchemalocation` que se seleccione.

COM y .NET

Firma: `SchemalocationHints(ENUMLoadSchemalocation 357 hint)`

Java

Firma: `public void setSchemalocationHints(ENUMLoadSchemalocation 357 hint)`

6.4.1.4.2.24 SchemaMapping

Establece qué asignación se debe usar para encontrar el esquema. La asignación se especifica por medio del literal de `ENUMSchemaMapping` que se seleccione.

COM y .NET

Firma: `SchemaMapping(ENUMSchemaMapping 359 mappingOption)`

 © 2017-2023 Altova GmbH

Java

Firma: `public void setSchemaMapping(ENUMSchemaMapping mappingOption)`

6.4.1.4.2.25 SchemaTextArray

Aporta el contenido de varios archivos de esquema XML. La entrada es una colección de cadenas, cada una de las cuales es el contenido del documento de esquema XML.

COM y .NET

Firma: `SchemaTextArray(object schemaDocs)`

Java

Firma: `public void setSchemaTextArray(object schemaDocs)`

6.4.1.4.2.26 Streaming

Habilita la validación por transmisión por secuencias. En el modo de transmisión por secuencias, los datos que están almacenados en memoria se minimizan y el procesamiento es más rápido. El valor `true` habilita la transmisión por secuencias y `false` la deshabilita. El valor predeterminado es `true`.

COM y .NET

Firma: `Streaming(boolean enable)`

Java

Firma: `public void setStreaming(boolean enable)`

6.4.1.4.2.27 XincludeSupport

Habilita o deshabilita el uso de los elementos `XInclude`. El valor `true` habilita la compatibilidad con XInclude y `false` la deshabilita. El valor predeterminado es `false`.

COM y .NET

Firma: `XincludeSupport(boolean xinclude)`

Java

Firma: `public void setXincludeSupport(boolean xinclude)`

6.4.1.4.2.28 XMLValidationMode

Establece el modo de validación XML, que es un literal de la enumeración ENUMXMLValidationMode[362] que determina si se debe comprobar la validez del documento o revisar su formato.

COM y .NET

Firma: `XMLValidationMode(ENUMXMLValidationMode[362] valMode)`

Java

Firma: `public void setXMLValidationMode(ENUMXMLValidationMode[362] valMode)`

6.4.1.4.2.29 XSDVersion

Establece la versión de XML Schema con la que se debe validar el documento XML. Es un literal de la enumeración ENUMXSDVersion[364].

COM y .NET

Firma: `XSDVersion(ENUMXSDVersion[364] version)`

Java

Firma: `public void setXSDVersion(ENUMXSDVersion[364] version)`

6.4.1.5 XQuery

La interfaz/clase **`IXQuery/XQuery`** ofrece métodos para (i) ejecutar documentos y actualizaciones XQuery y (ii) validar documentos relacionados con XQuery. También puede indicar datos para las ejecuciones mediante variables externas.

Tenga en cuenta que el nombre de la interfaz en la API COM/.NET es distinto que el de la clase en la API de Java:

 © 2017-2023 Altova GmbH

- En COM/.NET: **IXQuery**
- En Java: **XQuery**

6.4.1.5.1 Métodos

En esta sección se describen los métodos de la interfaz **IXQuery** (COM/.NET) y de la clase **XQuery** (Java).

6.4.1.5.1.1 AddExternalVariable

Añade el nombre y valor de una variable externa nueva. Cada variable externa y su valor se debe especificar en una llamada distinta al método. Las variables se deben declarar en el documento XQuery (y la declaración de tipo es opcional). Si el valor de la variable es una cadena de texto, ponga el valor entre comillas simples. El parámetro a`name`lmacena el nombre de la variable, que es un QName, como cadena. El parámetro `value` almacena el valor de la variable como cadena.

COM y .NET

Firma: **AddExternalVariable(**string **name**, string **value)**

Java

Firma: **public** void **addExternalVariable(**string **name**, string **value)**

6.4.1.5.1.2 ClearExternalVariableList

Borra la lista de variables externas creadas con el método AddExternalVariable[333].

COM y .NET

Firma: **ClearExternalVariableList()**

Java

Firma: **public** void **clearExternalVariableList()**

6.4.1.5.1.3 Execute

Ejecuta la transformación XQuery según la versión XQuery nombrada en la propiedad EngineVersion[337] y guarda el resultado en el archivo de salida nombrado en el parámetro `outputFile`. El parámetro es una cadena

que aporta la ubicación (ruta de acceso y nombre de archivo) del archivo de salida. El resultado es `true` si la transformación se ejecuta correctamente y `false` si no es así. Si se producen errores, se emite la excepción RaptorXMLException[312]. Use la propiedad `getLastErrorMessage` para obtener más información.

COM y .NET

Firma: `boolean Execute(string outputFile)`

Java

Firma: `public boolean Execute(string outputFile)`

6.4.1.5.1.4 ExecuteAndGetResultAsString

Ejecuta la transformación XQuery de acuerdo con la especificación XQuery Update nombrada en la propiedad EngineVersion[337] y devuelve el resultado en forma de cadena de texto. Este método no produce archivos de resultados adicionales como gráficos ni resultados secundarios. Tampoco almacena resultados binarios como archivos `.docx` OOXML. Si necesita más archivos de salida, utilice el método Execute[333].

COM y .NET

Firma: `string ExecuteAndGetResultAsString()`

Java

Firma: `public string executeAndGetResultAsString()`

6.4.1.5.1.5 ExecuteUpdate

Ejecuta la actualización XQuery de acuerdo con la especificación XQuery Update nombrada en la propiedad XQueryUpdateVersion[344] y guarda el resultado en el archivo de salida nombrado en el parámetro `outputFile`. El parámetro es una cadena que aporta la ubicación (ruta de acceso y nombre de archivo) del archivo de salida. El resultado es `true` si la actualización se ejecuta correctamente y `false` si no es así. Si se producen errores, se emite la excepción RaptorXMLException[312]. Use la propiedad `LastErrorMessage` para obtener más información.

COM y .NET

Firma: `boolean ExecuteUpdate(string outputFile)`

Java

 © 2017-2023 Altova GmbH

Firma: `public boolean executeUpdate(string outputFile)`

6.4.1.5.1.6 ExecuteUpdateAndGetResultAsString

Ejecuta la actualización XQuery de acuerdo con la especificación XQuery Update nombrada en la propiedad `XQueryUpdateVersion` [344] y devuelve el resultado en forma de cadena de texto. Este método no produce archivos de resultados adicionales como gráficos ni resultados secundarios. Tampoco almacena resultados binarios como archivos `.docx` OOXML.

COM y .NET

Firma: `string ExecuteUpdateAndGetResultAsString()`

Java

Firma: `public string executeUpdateAndGetResultAsString()`

6.4.1.5.1.7 IsValid

Devuelve el resultado de validar el documento XQuery de acuerdo con la especificación XQuery indicada en la propiedad `EngineVersion` [337]. El resultado es `true` si el documento es válido y `false` si no lo es. Si ocurre un error, se emite una excepción `RaptorXMLException` [312]. Use la propiedad `getLastErrorMessage` para obtener más información.

COM y .NET

Firma: `boolean IsValid()`

Java

Firma: `public boolean isValid()`

6.4.1.5.1.8 IsValidUpdate

Devuelve el resultado de validar el documento XQuery Update de acuerdo con la especificación XQuery Update indicada en la propiedad `XQueryUpdateVersion` [344]. El resultado es `true` si el documento es válido y `false` si no lo es. Si ocurre un error, se emite una excepción `RaptorXMLException` [312]. Use la propiedad `LastErrorMessage` para obtener más información.

COM y .NET

Firma: `boolean IsValidUpdate()`

Java

Firma: `public boolean isValidUpdate()`

6.4.1.5.2 Propiedades

En esta sección se describen las propiedades de la interfaz la propiedad (COM/.NET) y de la clase **`XQuery`** (Java).

6.4.1.5.2.1 AdditionalOutputs

Devuelve resultados adicionales del último trabajo que se ejecutó.

COM y .NET

Firma: `string AdditionalOutputs()`

Java

Firma: `public string getAdditionalOutputs()`

6.4.1.5.2.2 ChartExtensionsEnabled

Habilita o deshabilita las funciones de extensión de Altova para gráficos. El valor `true` habilita las extensiones para gráficos. El valor `false` las deshabilita. El valor predeterminado es `true`.

COM y .NET

Firma: `ChartExtensionsEnabled(boolean enable)`

Java

Firma: `public void setChartExtensionsEnabled(boolean enable)`

 © 2017-2023 Altova GmbH

6.4.1.5.2.3 DotNetExtensionsEnabled

Habilita o deshabilita las funciones de extensión .NET. El valor `true` habilita las extensiones .NET. El valor `false` las deshabilita. El valor predeterminado `true`.

COM y .NET

Firma: **`DotNetExtensionsEnabled(boolean enable)`**

Java

Firma: **`public void setDotNetExtensionsEnabled(boolean enable)`**

6.4.1.5.2.4 EngineVersion

Indica qué versión XQuery usar. El valor de la propiedad es un literal de ENUMXQueryVersion[364].

COM y .NET

Firma: **`EngineVersion(`**ENUMXQueryVersion[364] **`version)`**

Java

Firma: **`public void setEngineVersion(`**ENUMXQueryVersion[364] **`version)`**

6.4.1.5.2.5 IndentCharacters

Aporta la cadena de caracteres que se usará para aplicar sangría en el documento de salida.

COM y .NET

Firma: **`IndentCharacters(string indentChars)`**

Java

Firma: **`public void setIndentCharacters(string indentChars)`**

6.4.1.5.2.6 InputXMLFileName

Establece la ubicación, en forma de URL, del documento XML que se debe procesar. La cadena dada debe ser una URL absoluta que indique la ubicación exacta del archivo XML.

COM y .NET

Firma: `InputXMLFileName(string url)`

Java

Firma: `public void setInputXMLFileName(string url)`

6.4.1.5.2.7 InputXMLFromText

Aporta el contenido del documento XML que se debe procesar. La cadena dada es el contenido del documento XML que se debe procesar.

COM y .NET

Firma: `InputXMLFromText(string xml)`

Java

Firma: `public void setInputXMLFromText(string xml)`

6.4.1.5.2.8 JavaBarcodeExtensionLocation

Especifica la ubicación del archivo de extensión de código de barras. Para obtener más información consulte el apartado dedicado a las funciones de extensión de Altova para trabajar con códigos de barras 508. La cadena dada debe ser una URL absoluta que indique la ubicación base del archivo que se debe usar.

COM y .NET

Firma: `JavaBarcodeExtensionLocation(string url)`

Java

Firma: `public void setJavaBarcodeExtensionLocation(string url)`

 © 2017-2023 Altova GmbH

6.4.1.5.2.9 JavaExtensionsEnabled

Habilita o deshabilita las funciones de extensión Java. El valor `true` habilita las extensiones Java y `false` las deshabilita. El valor predeterminado es `true`.

COM y .NET

Firma: `JavaExtensionsEnabled(boolean enable)`

Java

Firma: `public void setJavaExtensionsEnabled(boolean enable)`

6.4.1.5.2.10 KeepFormatting

Especifica si se debe conservar el formato del documento original (en la medida de lo posible) o no. El valor `true` conserva el formato; `false` no conserva el formato. El valor predeterminado es `true`.

COM y .NET

Firma: `KeepFormatting(boolean keep)`

Java

Firma: `public void setKeepFormatting(boolean keep)`

6.4.1.5.2.11 LastErrorMessage

Recupera una cadena que es el último mensaje de error del motor de RaptorXML.

COM y .NET

Firma: `string LastErrorMessage()`

Java

Firma: `public string getLastErrorMessage()`

6.4.1.5.2.12 LoadXMLWithPSVI

Habilita la validación de archivos XML de entrada y genera información posterior a la validación con esquema sobre dichos archivos. El valor `true` habilita la validación XML y genera información posterior a la validación con esquema para los archivos XML. El valor `false` deshabilita la validación. El valor predeterminado es `true`.

COM y .NET

Firma: `LoadXMLWithPSVI(boolean enable)`

Java

Firma: `public void setLoadXMLWithPSVI(boolean enable)`

6.4.1.5.2.13 MainOutput

Devuelve el resultado principal del último trabajo que se ejecutó.

COM y .NET

Firma: `string MainOutput()`

Java

Firma: `public string getMainOutput()`

6.4.1.5.2.14 OutputEncoding

Establece la codificación del documento de salida. Use un nombre de codificación IANA oficial (p. ej. `UTF-8`, `UTF-16`, `US-ASCII`, `ISO-8859-1`) como cadena de texto.

COM y .NET

Firma: `OutputEncoding(string encoding)`

Java

Firma: `public void setOutputEncoding(string encoding)`

 © 2017-2023 Altova GmbH

6.4.1.5.2.15 OutputIndent

Habilita o deshabilita la sangría en el documento de salida. El valor `true` habilita la sangría; `false` la deshabilita.

COM y .NET

Firma: **`OutputIndent(boolean outputIndent)`**

Java

Firma: **`public void setOutputIndent(boolean outputIndent)`**

6.4.1.5.2.16 OutputMethod

Especifica la serialización del documento de salida. Los valores válidos son: `xml | xhtml | html | text`. El valor predeterminado es `xml`.

COM y .NET

Firma: **`OutputMethod(string format)`**

Java

Firma: **`public void setOutputMethod(string format)`**

6.4.1.5.2.17 OutputOmitXMLDeclaraton

Habilita/deshabilita la inclusión de la declaración XML en el documento del resultado. El valor `true` omite la declaración; `false` la incluye. El valor predeterminado es `false`.

COM y .NET

Firma: **`OutputOmitXMLDeclaration(boolean omitDeclaration)`**

Java

Firma: **`public void setOutputOmitXMLDeclaration(boolean omitDeclaration)`**

6.4.1.5.2.18 UpdatedXMLWriteMode

Especifica cómo se deben gestionar las actualizaciones en el archivo XML. El valor de la propiedad es un literal de ENUMXQueryUpdatedXML[363].

COM y .NET

Firma: **UpdateXMLWriteMode(**ENUMXQueryUpdatedXML[363] **updateMode)**

Java

Firma: **public** void **setUpdateXMLWriteMode(**ENUMXQueryUpdatedXML[363] **updateMode)**

6.4.1.5.2.19 XincludeSupport

Habilita o deshabilita el uso de los elementos `XInclude`. El valor `true` habilita la compatibilidad con XInclude y `false` la deshabilita. El valor predeterminado es `false`.

COM y .NET

Firma: **XincludeSupport(**boolean **xinclude)**

Java

Firma: **public** void **setXincludeSupport(**boolean **xinclude)**

6.4.1.5.2.20 XMLValidationErrorsAsWarnings

Define si los errores de validación XML se tratan como advertencias o no. Toma el valor booleano `true` o `false`.

COM y .NET

Firma: **XMLValidationErrorsAsWarnings(**boolean **enable)**

Java

Firma: **public** void **setXMLValidationErrorsAsWarnings(**boolean **enable)**

 © 2017-2023 Altova GmbH

6.4.1.5.2.21 XMLValidationMode

Establece el modo de validación XML, que es un literal de la enumeración ENUMXMLValidationMode[362] que determina si se debe comprobar la validez del documento o revisar su formato.

COM y .NET

Firma: **XMLValidationMode(**ENUMXMLValidationMode[362] **valMode)**

Java

Firma: **public** void **setXMLValidationMode(**ENUMXMLValidationMode[362] **valMode)**

6.4.1.5.2.22 XQueryFileName

Indica qué archivo XQuery se debe usar. La cadena indicada debe ser una URL absoluta que dé la ubicación del archivo XQuery que se debe usar.

COM y .NET

Firma: **XQueryFileName(**string **fileurl)**

Java

Firma: **public** void **setXQueryFileName(**string **fileurl)**

6.4.1.5.2.23 XQueryFromText

Suministra, como cadena de texto, el contenido del archivo XQuery que se debe usar.

COM y .NET

Firma: **XQueryFromText(**string **xqtext)**

Java

Firma: **public** void **setXQueryFromText(**string **xqtext)**

6.4.1.5.2.24 XQueryUpdateVersion

Indica qué versión XQuery Update usar. El valor de la propiedad es un literal de ENUMXQueryVersion[364].

COM y .NET

Firma: **XQueryUpdateVersion(**ENUMXQueryVersion[364] **version)**

Java

Firma: **public** void **setXQueryUpdateVersion(**ENUMXQueryVersion[364] **version)**

6.4.1.5.2.25 XSDVersion

Establece la versión de XML Schema con la que se debe validar el documento XML. Es un literal de la enumeración ENUMXSDVersion[364].

COM y .NET

Firma: **XSDVersion(**ENUMXSDVersion[364] **version)**

Java

Firma: **public** void **setXSDVersion(**ENUMXSDVersion[364] **version)**

6.4.1.6 XSLT

La interfaz/clase **IXSLT/XSLT** ofrece métodos para ejecutar transformaciones XSLT y validar documentos relacionados con XSLT. También puede aportar datos para las transformaciones mediante parámetros externos.

Tenga en cuenta que el nombre de la interfaz en la API COM/.NET es distinto que el de la clase en la API de Java:

- En COM/.NET: **IXSLT**
- En Java: **XSLT**

© 2017-2023 Altova GmbH

6.4.1.6.1 Métodos

En esta sección se describen los métodos de la interfaz **IXSLT** (COM/.NET) y de la clase **XSLT** (Java).

6.4.1.6.1.1 AddExternalParameter

Añade el nombre y valor de un parámetro externo nuevo. Cada parámetro externo y su valor debe especificarse en una llamada distinta al método. Los parámetros deben declararse en el documento XSLT. Como los valores de parámetros son expresiones XPath, los valores de parámetros que sean cadenas deben ir entre comillas simples. El parámetro `name` almacena el nombre de la variable, que es un QName, como cadena de texto. El parámetro `value` almacena el valor de la variable como cadena de texto.

COM y .NET

Firma: **AddExternalParameter(**string **name,** string **value)**

Java

Firma: **public** void **addExternalParameter(**string **name,** string **value)**

6.4.1.6.1.2 ClearExternalParameterList

Borra la lista de parámetros externos creada con el método AddExternalParameter 345.

COM y .NET

Firma: **ClearExternalParameterList()**

Java

Firma: **public** void **clearExternalParameterList()**

6.4.1.6.1.3 Execute

Ejecuta la transformación XSLT de acuerdo con la especificación XSLT indicada en la propiedad EngineVersion 348 y guarda el resultado en el archivo de salida indicado en el parámetro `outputFile`. Si ocurre un error, se emite una excepción RaptorXMLException 312. Use la propiedad `LastErrorMessage` para obtener más información.

COM y .NET

Firma: `boolean Execute(string outputFile)`

Java

Firma: `public boolean execute(string outputFile)`

6.4.1.6.1.4 ExecuteAndGetResultAsString

Ejecuta la transformación XSLT de acuerdo con la especificación XSLT indicada en la propiedad `EngineVersion`[348] y devuelve el resultado en forma de cadena de texto. Este método no produce archivos de resultados adicionales como gráficos ni resultados secundarios. Tampoco almacena resultados binarios como archivos `.docx` OOXML. Si necesita más archivos de salida, utilice el método `Execute`[345]. Si ocurre un error, se emite una excepción `RaptorXMLException`[312]. Use la propiedad `LastErrorMessage` para obtener más información.

COM y .NET

Firma: `string ExecuteAndGetResultAsString()`

Java

Firma: `public string executeAndGetResultAsString()`

6.4.1.6.1.5 ExecuteAndGetResultAsStringWithBaseOutputURI

Ejecuta la transformación XSLT de acuerdo con la especificación XSLT indicada en la propiedad `EngineVersion`[348] y devuelve el resultado en forma de cadena de texto en la ubicación definida por el URI base. El parámetro `baseURI` es una cadena que aporta un URI. Este método no produce archivos de resultados adicionales como gráficos ni resultados secundarios. Tampoco almacena resultados binarios como archivos `.docx` OOXML.Si necesita más archivos de salida, utilice el método `Execute`[345]. Si ocurre un error, se emite una excepción `RaptorXMLException`[312]. Use la propiedad `LastErrorMessage` para obtener más información.

COM y .NET

Firma: `string ExecuteAndGetResultAsStringWithBaseOutputURI(string baseURI)`

Java

 © 2017-2023 Altova GmbH

Firma: `public string ExecuteAndGetResultAsStringWithBaseOutputURI(string baseURI)`

6.4.1.6.1.6 IsValid

Devuelve el resultado de validar el documento XSLT de acuerdo con la especificación XSLT indicada en la propiedad `EngineVersion`[348]. El resultado es `true` si el documento es válido y `false` si no lo es. Si ocurre un error, se emite una excepción `RaptorXMLException`[312]. Use el método `LastErrorMessage` para obtener más información.

COM y .NET

Firma: `boolean IsValid()`

Java

Firma: `public boolean isValid()`

6.4.1.6.2 Propiedades

En esta sección se describen las propiedades de la interfaz **IXSLT** (COM/.NET) y de la clase **IXSLT** (Java).

6.4.1.6.2.1 AdditionalOutputs

Devuelve resultados adicionales del último trabajo que se ejecutó.

COM y .NET

Firma: `string AdditionalOutputs()`

Java

Firma: `public string getAdditionalOutputs()`

6.4.1.6.2.2 ChartExtensionsEnabled

Habilita o deshabilita las funciones de extensión de Altova para gráficos. El valor `true` habilita las extensiones para gráficos. El valor `false` las deshabilita. El valor predeterminado es `true`.

COM y .NET

Firma: **ChartExtensionsEnabled(**boolean **enable)**

Java

Firma: **public** void **setChartExtensionsEnabled(**boolean **enable)**

6.4.1.6.2.3 DotNetExtensionsEnabled

Habilita o deshabilita las funciones de extensión .NET. El valor `true` habilita las extensiones .NET. El valor `false` las deshabilita. El valor predeterminado `true`.

COM y .NET

Firma: **DotNetExtensionsEnabled(**boolean **enable)**

Java

Firma: **public** void **setDotNetExtensionsEnabled(**boolean **enable)**

6.4.1.6.2.4 EngineVersion

Indica qué versión XSLT se debe usar. El valor de la propiedad es un literal de ENUMXSLTVersion[365].

COM y .NET

Firma: **EngineVersion(**ENUMXSLTVersion[365] **version)**

Java

Firma: **public** void **setEngineVersion(**ENUMXSLTVersion[365] **version)**

6.4.1.6.2.5 IndentCharacters

Aporta la cadena de caracteres que se usará para aplicar sangría en el documento de salida.

COM y .NET

 © 2017-2023 Altova GmbH

Firma: `IndentCharacters(string indentChars)`

Java

Firma: `public void setIndentCharacters(string indentChars)`

6.4.1.6.2.6 InitialTemplateMode

Establece el modo inicial para el procesamiento XSLT. Se procesarán aquellas plantillas cuyo valor de modo sea igual a la cadena indicada.

COM y .NET

Firma: `InitialTemplateMode(string mode)`

Java

Firma: `public void setInitialTemplateMode(string mode)`

6.4.1.6.2.7 InputXMLFileName

Establece la ubicación, en forma de URL, del documento XML que se debe procesar. La cadena dada debe ser una URL absoluta que indique la ubicación exacta del archivo XML.

COM y .NET

Firma: `InputXMLFileName(string url)`

Java

Firma: `public void setInputXMLFileName(string url)`

6.4.1.6.2.8 InputXMLFromText

Aporta el contenido del documento XML que se debe procesar. La cadena dada es el contenido del documento XML que se debe procesar.

COM y .NET

Firma: `InputXMLFromText(string xml)`

Java

Firma: `public void setInputXMLFromText(string xml)`

6.4.1.6.2.9 JavaBarcodeExtensionLocation

Especifica la ubicación del archivo de extensión de código de barras. Para obtener más información consulte el apartado dedicado a las funciones de extensión de Altova para trabajar con códigos de barras [508]. La cadena dada debe ser una URL absoluta que indique la ubicación base del archivo que se debe usar.

COM y .NET

Firma: `JavaBarcodeExtensionLocation(string url)`

Java

Firma: `public void setJavaBarcodeExtensionLocation(string url)`

6.4.1.6.2.10 JavaExtensionsEnabled

Habilita o deshabilita las funciones de extensión Java. El valor `true` habilita las extensiones Java y `false` las deshabilita. El valor predeterminado es `true`.

COM y .NET

Firma: `JavaExtensionsEnabled(boolean enable)`

Java

Firma: `public void setJavaExtensionsEnabled(boolean enable)`

6.4.1.6.2.11 LastErrorMessage

Recupera una cadena que es el último mensaje de error del motor de RaptorXML.

COM y .NET

Firma: `string LastErrorMessage()`

 © 2017-2023 Altova GmbH

Java

Firma: `public string getLastErrorMessage()`

6.4.1.6.2.12 LoadXMLWithPSVI

Habilita la validación de archivos XML de entrada y genera información posterior a la validación con esquema sobre dichos archivos. El valor `true` habilita la validación XML y genera información posterior a la validación con esquema para los archivos XML. El valor `false` deshabilita la validación. El valor predeterminado es `true`.

COM y .NET

Firma: `LoadXMLWithPSVI(boolean enable)`

Java

Firma: `public void setLoadXMLWithPSVI(boolean enable)`

6.4.1.6.2.13 MainOutput

Devuelve el resultado principal del último trabajo que se ejecutó.

COM y .NET

Firma: `string MainOutput()`

Java

Firma: `public string getMainOutput()`

6.4.1.6.2.14 NamedTemplateEntryPoint

Especifica el nombre, como cadena de texto, de la plantilla con nombre que debe utilizarse como punto de entrada de la transformación

COM y .NET

Firma: `NamedTemplateEntryPoint(string template)`

Java

Firma: `public void setNamedTemplateEntryPoint(string template)`

6.4.1.6.2.15 SchemaImports

Especifica cómo se deben realizar las importaciones de esquema, en función de los valores de atributo de los elementos `xs:import`. El tipo de importación se especifica con el literal de `ENUMSchemaImports` que se aporta.

COM y .NET

Firma: `SchemaImports(ENUMSchemaImports 358 importOption)`

Java

Firma: `public void setSchemaImports(ENUMSchemaImports 358 importOption)`

6.4.1.6.2.16 SchemalocationHints

Especifica el mecanismo que se debe utilizar para encontrar el esquema. El mecanismo se especifica por medio del literal de `ENUMLoadSchemalocation` que se seleccione.

COM y .NET

Firma: `SchemalocationHints(ENUMLoadSchemalocation 357 hint)`

Java

Firma: `public void setSchemalocationHints(ENUMLoadSchemalocation 357 hint)`

6.4.1.6.2.17 SchemaMapping

Establece qué asignación se debe usar para encontrar el esquema. La asignación se especifica por medio del literal de `ENUMSchemaMapping` que se seleccione.

COM y .NET

Firma: `SchemaMapping(ENUMSchemaMapping 359 mappingOption)`

 © 2017-2023 Altova GmbH

Java

Firma: `public void setSchemaMapping(ENUMSchemaMapping[359] mappingOption)`

6.4.1.6.2.18 StreamingSerialization

Habilita la serialización de secuencias de datos. En el modo de transmisión por secuencias, los datos almacenados en memoria se minimizan y el procesamiento es más rápido. El valor `true` habilita la serialización de secuencias de datos; `false` la deshabilita.

COM y .NET

Firma: `StreamingSerialization(boolean enable)`

Java

Firma: `public void setStreamingSerialization(boolean enable)`

6.4.1.6.2.19 XincludeSupport

Habilita o deshabilita el uso de los elementos `XInclude`. El valor `true` habilita la compatibilidad con XInclude y `false` la deshabilita. El valor predeterminado es `false`.

COM y .NET

Firma: `XincludeSupport(boolean xinclude)`

Java

Firma: `public void setXincludeSupport(boolean xinclude)`

6.4.1.6.2.20 XMLValidationErrorsAsWarnings

Define si los errores de validación XML se tratan como advertencias o no. Toma el valor booleano `true` o `false`.

COM y .NET

Firma: `XMLValidationErrorsAsWarnings(boolean enable)`

Java

Firma: `public void setXMLValidationErrorsAsWarnings(boolean enable)`

6.4.1.6.2.21 XMLValidationMode

Establece el modo de validación XML, que es un literal de la enumeración ENUMXMLValidationMode[362] que determina si se debe comprobar la validez del documento o revisar su formato.

COM y .NET

Firma: `XMLValidationMode(ENUMXMLValidationMode[362] valMode)`

Java

Firma: `public void setXMLValidationMode(ENUMXMLValidationMode[362] valMode)`

6.4.1.6.2.22 XSDVersion

Establece la versión de XML Schema con la que se debe validar el documento XML. Es un literal de la enumeración ENUMXSDVersion[364].

COM y .NET

Firma: `XSDVersion(ENUMXSDVersion[364] version)`

Java

Firma: `public void setXSDVersion(ENUMXSDVersion[364] version)`

6.4.1.6.2.23 XSLFileName

Especifica qué archivo XSLT usar. La cadena indicada debe ser una URL absoluta que dé la ubicación del archivo XSLT que se debe usar.

COM y .NET

Firma: `XSLFileName(string fileurl)`

© 2017-2023 Altova GmbH

Java

Firma: `public void setXSLFileName(string fileurl)`

6.4.1.6.2.24 XSLFromText

Ofrece, en forma de cadena de texto, el contenido del documento XSLT que se debe usar.

COM y .NET

Firma: `XSLFromText(string xsltext)`

Java

Firma: `public void setXSLFromText(string xsltext)`

6.4.2 Enumeraciones

En esta sección se describen las enumeraciones de los servidores API COM/.NET y Java. Cada descripción incluye un enlace a los métodos o las propiedades que usa esa enumeración.

- ENUMAssessmentMode 355
- ENUMErrorFormat 356
- ENUMLoadSchemalocation 357
- ENUMSchemaImports 358
- ENUMSchemaMapping 359
- ENUMValidationType 360
- ENUMWellformedCheckType 361
- ENUMXMLValidationMode 362
- ENUMXQueryUpdatedXML 363
- ENUMXQueryVersion 364
- ENUMXSDVersion 364
- ENUMXSLTVersion 365

6.4.2.1 ENUMAssessmentMode

Contiene literales de enumeración que definen si el modo de evaluación del validador XML debe ser estricto o laxo:

- **`eAssessmentModeStrict`**: establece el modo de evaluación de la validez del esquema en `Strict`. Es el valor predeterminado.

- **eAssessmentModeLax**: establece el modo de evaluación de la validez del esquema en Lax.

COM y .NET

eAssessmentModeStrict	= 0
eAssessmentModeLax	= 1

Usada por

Interfaz	*Propiedad*
IXMLValidator 320	AssessmentMode 323

Java

```
public enum ENUMAssessmentMode {
        eAssessmentModeLax
        eAssessmentModeStrict    }
```

Usada por

Clase	*Método*
XMLValidator 320	setAssessmentMode 323

6.4.2.2 ENUMErrorFormat

Contiene literales de enumeración que especifican el formato de los errores de salida:

- **eFormatText**: establece el formato de los errores de salida en Text. Es el valor predeterminado.
- **eFormatShortXML**: establece el formato de los errores de salida en ShortXML. Este formato es una versión abreviada del formato LongXML.
- **eFormatLongXML**: establece el formato de los errores de salida en LongXML. Este formato es el que ofrece información más detallada.

COM y .NET

eFormatText	= 0
eFormatShortXML	= 1
eFormatLongXML	= 2

Usada por

Interfaz	*Propiedad*
IServer 303	ErrorFormat 307

 © 2017-2023 Altova GmbH

Java

```
public enum ENUMErrorFormat {
       eFormatText
       eFormatShortXML
       eFormatLongXML }
```

Usada por

Clase	*Método*
RaptorXMLFactory 303	setErrorFormat 307

6.4.2.3 ENUMLoadSchemalocation

Contiene literales de enumeración que indican cómo determinar la ubicación del esquema. Para ello se basa en el atributo de ubicación del esquema del documento de instancia XML. Este atributo puede ser `xsi:schemaLocation` o `xsi:noNamespaceSchemaLocation`.

- **`eSHLoadBySchemalocation`** usa la URL del atributo de ubicación del esquema en el documento de instancia XML. Este literal de enumeración es el **valor predeterminado**.
- **`eSHLoadByNamespace`** usa la parte de espacio de nombres del atributo `xsi:schemaLocation` (en el caso de `xsi:noNamespaceSchemaLocation` es una cadena vacía) y busca el esquema a través de la asignación de catálogo.
- **`eSHLoadCombiningBoth`**: si la URL de espacio de nombres o la de ubicación del esquema tienen una asignación de catálogo, esta asignación de catálogo se utiliza. Si ambas tienen una asignación de catálogo, entonces es el valor del parámetro ENUMSchemaMapping 359 lo que decide cuál de las asignaciones se utiliza. Si ninguna tiene una asignación de catálogo, se usa la URL.
- **`eSHLoadIgnore`**: se ignoran los atributos `xsi:schemaLocation` y `xsi:noNamespaceSchemaLocation`.

COM y .NET

eSHLoadBySchemalocation	= 0
eSHLoadByNamespace	= 1
eSHLoadCombiningBoth	= 2
eSHLoadIgnore	= 3

Usada por

Interfaz	*Propiedad*
IXMLValidator 320	SchemalocationHints 330
IXSLT 344	SchemalocationHints 352

Java

```
public enum ENUMLoadSchemalocation {
       eSHLoadBySchemalocation
       eSHLoadByNamespace
       eSHLoadCombiningBoth
       eSHLoadIgnore    }
```

Usada por

Clase	*Método*
XMLValidator 320	setSchemalocationHints 330
XSLT 344	setSchemalocationHints 352

6.4.2.4 ENUMSchemaImports

Contiene los literales de enumeración que definen el comportamiento de los elementos `xs:import`, cada uno de los cuales tiene los atributos opcionales `namespace` y `schemaLocation`.

- **`eSILoadBySchemalocation`** usa el valor del atributo `schemaLocation` para buscar el esquema, teniendo en cuenta las asignaciones de catálogo. Si está presente el atributo `namespace`, se importa el espacio de nombres (con licencia).
- **`eSILoadPreferringSchemalocation`**: si está presente el atributo `schemaLocation`, se utiliza teniendo en cuenta las asignaciones del catálogo. Si no está presente el atributo `schemaLocation`, se usa el valor del atributo `namespace` a través de una asignación de catálogo. Este literal es el **valor predeterminado** de la enumeración.
- **`eSILoadByNamespace`** usa el valor del atributo `namespace` para encontrar el esquema a través de una asignación de catálogo.
- **`eSILoadCombiningBoth`**: si el atributo `namespace` o `schemaLocation` tienen una asignación de catálogo, esta asignación de catálogo se utiliza. Si ambos tienen una asignación de catálogo, entonces es el valor del parámetro `ENUMSchemaMapping` 359 lo que decide cuál de las asignaciones se utiliza. Si ninguna tiene una asignación de catálogo, se usa el valor del atributo `schemaLocation` (que debería ser una URL).
- **`eSILicenseNamespaceOnly`**: el espacio de nombres se importa. No se importa ningún documento de esquema.

COM y .NET

eSILoadBySchemalocation	= 0
eSILoadPreferringSchemalocation	= 1
eSILoadByNamespace	= 2
eSICombiningBoth	= 3

© 2017-2023 Altova GmbH

eSILicenseNamespaceOnly	= 4

Usada por

Interfaz	*Propiedad*
IXMLValidator 320	SchemaImports 330
IXSLT 344	SchemaImports 352

Java

```
public enum ENUMSchemaImports {
        eSILoadBySchemalocation
        eSILoadPreferringSchemalocation
        eSILoadByNamespace
        eSILoadCombiningBoth
        eSILicenseNamespaceOnly    }
```

Usada por

Clase	*Método*
XMLValidator 320	setSchemaImports 330
XSLT 344	setSchemaImports 352

6.4.2.5 ENUMSchemaMapping

Contiene los literales de enumeración que definen cuál de las dos asignaciones de catálogo se prefiere. Esta enumeración sirve para eliminar ambigüedades en ENUMLoadSchemalocation 357 y ENUMSchemaImports 358.

- **eSMPreferNamespace**: establece que se debe seleccionar el espacio de nombres.
- **eSMPreferSchemalocation**: establece que se debe seleccionar la ubicación del esquema. Es el valor predeterminado.

COM y .NET

eSMPreferSchemalocation	= 0
eSMPreferNamespace	= 1

Usada por

Interfaz	*Propiedad*
IXMLValidator 320	SchemaMapping 330
IXSLT 344	SchemaMapping 352

Java

```
public enum ENUMSchemaMapping {
       eSMPreferSchemalocation
       eSMPreferNamespace    }
```

Usada por

Clase	*Método*
IXMLValidator 320	setSchemaMapping 330
IXSLT 344	setSchemaMapping 352

6.4.2.6 ENUMValidationType

Contiene el literal de enumeración que especifica qué tipo de validación se debe llevar a cabo y, en el caso de los documentos XML, si se usa una DTD o un XSD para la validación.

- **eValidateAny**: el tipo de documento (por ejemplo, XML o XSD) se detecta automáticamente y en base a este se especifica el tipo de validación.
- **eValidateXMLWithDTD**: especifica que el documento XML debe validarse con un documento DTD.
- **eValidateXMLWithXSD**: especifica que el documento XML debe validarse con un documento XSD (esquema XML).
- **eValidateDTD**: especifica que debe validarse un documento DTD.
- **eValidateXSD**: especifica que debe validarse un documento XSD (W3C XMLSchema).
- **eValidateJSON**: especifica que debe validarse un documento de instancia JSON.
- **eValidateJSONSchema**: especifica que debe validarse un esquema JSON de acuerdo con las normas de JSON Schema v 4.
- **eValidateAvro**: especifica que debe validarse un archivo binario Avro. Los datos Avro del archivo binario se validan con el esquema Avro que contiene el archivo binario.
- **eValidateAvroSchema**: especifica que debe validarse un esquema Avro según las normas de la especificación Avro schema.
- **eValidateAvroJSON**: especifica la validación de un documento de datos Avro en serialización JSON con un esquema Avro.

COM y .NET

eValidateAny	= 0
eValidateXMLWithDTD	= 1
eValidateXMLWithXSD	= 2
eValidateDTD	= 3
eValidateXSD	= 4
eValidateJSON	= 5

 © 2017-2023 Altova GmbH

eValidateJSONSchema	= 6
eValidateAvro	= 7
eValidateAvroSchema	= 8
eValidateAvroJSON	= 9

Usada por

Interfaz	*Método*
IXMLValidator 320	isValid 322

Java

```
public enum ENUMValidationType {
        eValidateAny
        eValidateXMLWithDTD
        eValidateXMLWithXSD
        eValidateDTD
        eValidateXSD
        eValidateJSON
        eValidateJSONSchema
        eValidateAvro
        eValidateAvroSchema
        eValidateAvroJSON }
```

Usada por

Clase	*Método*
XMLValidator 320	isValid 322

6.4.2.7 ENUMWellformedCheckType

Contiene los literales de enumeración que definen el tipo de documento cuyo formato debe comprobarse (XML, DTD o JSON).

- **eWellformedAny**: el tipo de documento y de validación se detectan automáticamente.
- **eWellformedXML**: comprueba si un documento XML tiene el formato correcto.
- **eWellformedDTD**: comprueba si un documento DTD tiene el formato correcto.
- **eWellformedJSON**: comprueba si un documento JSON tiene el formato correcto.

COM y .NET

eWellFormedAny	= 0
eWellFormedXML	= 1
eWellFormedDTD	= 2

eWellFormedJSON	= 3

Usada por

Interfaz	*Método*
IXMLValidator 320	isWellFormed 322

Java

```
public enum ENUMWellformedCheckType {
        eWellformedAny
        eWellformedXML
        eWellformedDTD
        eWellformedJSON    }
```

Usada por

Clase	*Método*
XMLValidator 320	isWellFormed 322

6.4.2.8 ENUMXMLValidationMode

Contiene literales de enumeración que definen el modo de validación XML que se debe usar (validación o comprobación de formato).

- **eProcessingModeWF**: establece que el modo de procesamiento XML es `Wellformed`. Es el valor predeterminado.
- **eProcessingModeValid**: establece que el modo de procesamiento XML es `Validation`.

eProcessingModeID: solo para uso interno.

COM y .NET

eXMLValidationModeWF	= 0
eXMLValidationModeID	= 1
eXMLValidationModeValid	= 2

Usada por

Interfaz	*Propiedad*
IXMLValidator 320	XMLValidationMode 332
IXQuery 332	XMLValidationMode 343
IXSLT 344	XMLValidationMode 354

© 2017-2023 Altova GmbH

Java

```
public enum ENUMXMLValidationMode {
       eProcessingModeValid
       eProcessingModeWF
       eProcessingModeID  }
```

Usada por

Clase	*Método*
XMLValidator 320	setXMLValidationMode 332
XQuery 332	setXMLValidationMode 343
XSLT 344	setXMLValidationMode 354

6.4.2.9 ENUMXQueryUpdatedXML

Determina cómo se deben gestionar las actualizaciones XQuery.

- **eUpdatedDiscard**: las actualizaciones se descartan y no se escriben en el archivo.
- **eUpdatedWriteback**: las actualizaciones se escriben en el archivo de entrada indicado con (set) InputXMLFileName 338.
- **eUpdatedAsMainResult**: las actualizaciones se escriben en la ubicación indicada por el parámetro outputFile de ExecuteUpdate 314.

COM y .NET

eUpdatedDiscard	= 1
eUpdatedWriteback	= 2
eUpdatedAsMainResult	= 3

Usada por

Interfaz	*Propiedad*
IXQuery 332	UpdatedXMLWriteMode 342

Java

```
public enum ENUMXQueryUpdatedXML {
       eUpdatedDiscard
       eUpdatedWriteback
       eeUpdatedAsMainResult }
```

Usada por

Clase	*Método*
XQuery 332	setUpdatedXMLWriteMode 342

6.4.2.10 ENUMXQueryVersion

Contiene literales de enumeración que especifican qué versión de XQuery se debe usar (ejecución o validación).

- **eXQVersion10**: establece que la versión XQuery es XQuery 1.0.
- **eXQVersion30**: establece que la versión XQuery es XQuery 3.0. Es el valor predeterminado.
- **eXQVersion31**: establece que la versión XQuery es XQuery 3.1.

Note: The Java enumeration literals are differently named than the COM/.NET literals. *See below.*

COM y .NET

eXQVersion10	= 1
eXQVersion30	= 3
eXQVersion31	= 31

Usada por

Interfaz	*Propiedad*
IXQuery 332	EngineVersion 337

Java

```
public enum ENUMXQueryVersion {
       eVersion10
       eVersion30
       eVersion31 }
```

Usada por

Clase	*Método*
XQuery 332	setEngineVersion 337

6.4.2.11 ENUMXSDVersion

Specifies the XML Schema version to use for validation.

- **eXSDVersionAuto**: la versión XSD se detecta automáticamente tras analizar el documento XSD. Si el atributo `vc:minVersion` del documento XSD tiene el valor `1.1`, se considera que el documento tiene

 © 2017-2023 Altova GmbH

la versión XSD `1.1`. Si este atributo tiene cualquier otro valor o si el atributo no está en el documento, se entiende que el documento tiene la versión XSD 1.0.

- **`eXSDVersion10`**: establece que la versión de XML Schema que se debe usar es XML Schema 1.0.
- **`eXSDVersion11`**: establece que la versión de XML Schema que se debe usar es XML Schema 1.1.

COM y .NET

`eXSDVersionAuto`	`= 0`
`eXSDVersion10`	`= 1`
`eXSDVersion11`	`= 2`

Usada por

Interfaz	*Propiedad*
IXMLValidator 320	XSDVersion 332
IXQuery 332	XSDVersion 344
IXSLT 344	XSDVersion 354

Java

```
public enum ENUMXSDVersion {
       eXSDVersionAuto
       eXSDVersion10
       eXSDVersion11   }
```

Usada por

Clase	*Método*
XMLValidator 320	setXSDVersion 332
XQuery 332	setXSDVersion 344
XSLT 344	setXSDVersion 354

6.4.2.12 ENUMXSLTVersion

Contiene literales de enumeración que especifican qué versión XSLT se debe usar para el procesamiento (validación o transformación XSLT).

- **`eVersion10`**: establece que la versión XSLT que se debe usar es XSLT 1.0.
- **`eVersion20`**: establece que la versión XSLT que se debe usar es XSLT 2.0.
- **`eVersion30`**: establece que la versión XSLT que se debe usar es XSLT 3.0.

COM y .NET

eVersion10	= 1
eVersion20	= 2
eVersion30	= 3

Usada por

Interfaz	*Propiedad*
IXSLT 344	EngineVersion 348

Java

```
public enum ENUMXSLTVersion {
        eVersion10
        eVersion20
        eVersion30 }
```

Usada por

Clase	*Método*
XSLT 344	setEngineVersion 348

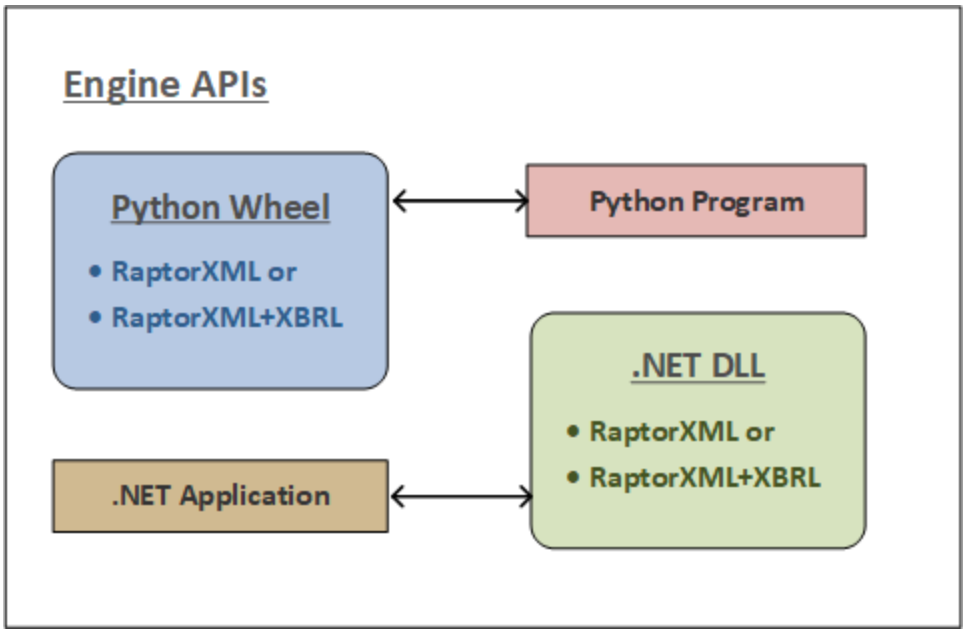
© 2017-2023 Altova GmbH

7 APIs de motor: Python y .NET

RaptorXML Server viene con dos APIs de motor:

- un archivo wheel Python (`.whl`), que es la API de motor de Python: `raptorxml<versiondetails>.whl`
- un archivo .NET DLL (`.dll`), que es la API de motor .NET: `raptorxmlapi.dll`

Estas dos APIs de motor permiten acceder a las funciones y características de RaptorXML Server como paquetes autónomos e independientes de RaptorXML Server (*véase imagen siguiente*). Es necesario instalar estos paquetes en el equipo del usuario para después poder importarlos como módulos de Python o integrarlos en una aplicación .NET personalizada. Gracias a que todo el procesamiento tiene lugar de forma local en el equipo del usuario, las APIs de motor de Python y .NET ofrecen acceso detallado a los modelos de datos de instancias XML o XBRL, esquemas XSD y taxonomías XBRL, siempre que sean válidos. Las APIs exponen un amplio conjunto de métodos para iterar el contenido de instancias XBRL u obtener información específica de taxonomías XBRL con unas líneas de código.



Tenga en cuenta estos puntos con respecto a las APIs de motor:

- Una vez haya instalado RaptorXML Server, encontrará las dos APIs de motor en la carpeta `bin` de la carpeta de instalación de RaptorXML Server.
- Las APIs de motor ofrecen un procesamiento adicional superior gracias a los objetos más versátiles de sus APIs.
- Para poder usar una API de motor es necesario tener instalada una versión de RaptorXML Server con licencia en el equipo en el que se vayan a ejecutar el programa en Python o la aplicación en .NET (*véase Utilización, más abajo*).

Uso

Puede crear un programa en Python o una aplicación en .NET como sigue:

Programa en Python
Un programa en Python puede acceder a las funciones de RaptorXML usando objetos de la API de Python[371] (*véase aquí*[371]). Cuando se ejecute el programa Python, este usará la biblioteca de RaptorXML que se instaló en su entorno Python al instalar el archivo wheel de Python. Tenga en cuenta que esta **solo** es compatible con la versión 3.11.5 de Python.

Aplicación en .NET
Una aplicación .NET puede acceder a las funciones de RaptorXML usando objetos de la API de .NET[381] (*véase aquí*[381]). Cuando se ejecute la aplicación .NET, esta usará el RaptorXML que se encuentre en el DLL de la API de .NET.

Licencias

Para poder usar una API de motor es necesario que se haya instalado una versión de RaptorXML Server con licencia en el equipo en el que se vayan a ejecutar el programa en Python o la aplicación en .NET (*consulte el apartado Licencias*[369] *para más información*).

© 2017-2023 Altova GmbH

7.1 Licencias

Para poder ejecutar el paquete de la API en un equipo cliente, deberá asignarle una licencia como si se tratase de un cliente RaptorXML Server. La asignación de licencias es un proceso de dos pasos:

1. Primero debe registrar el equipo como cliente RaptorXML Server con el servidor de licencias Altova LicenseServer.
2. Después debe asignar al equipo una licencia de RaptorXML Server desde Altova LicenseServer.

Si tiene pensado ejecutar el paquete de la API desde un equipo concreto, tiene dos opciones:

- Si el equipo cliente ya ejecuta una versión con licencia de RaptorXML Server, entonces podrá ejecutar el paquete de la API sin necesidad de configurar nada. El motivo es que el equipo ya tiene asignada una licencia que le permite ejecutar RaptorXML Server. Por tanto, el uso del paquete de la API en dicho equipo está habilitado por medio de la licencia que se asignó a RaptorXML Server en dicho equipo.
- Si RaptorXML Server no está instalado en el equipo y prefiere no instalarlo por el motivo que sea, deberá registrar el equipo como cliente RaptorXML Server y asignarle una licencia de RaptorXML Server (ver instrucciones más abajo).

Para registrar un equipo (que no tenga RaptorXML Server instalado) como cliente RaptorXML Server basta con usar la aplicación de la línea de comandos **`registerlicense.exe`**, disponible en la carpeta **`bin`** de la aplicación.

Windows	`Archivos de programa\Altova\RaptorXMLServer2024\bin`
Linux	`/opt/Altova/RaptorXMLServer2024/bin`
Mac	`/usr/local/Altova/RaptorXMLServer2024/bin`

Después, ejecute el comando:

```
registerlicense <LicenseServer>
```

donde **`<LicenseServer>`** será la dirección IP o nombre de host del equipo donde se ejecuta el servidor de licencias LicenseServer.

Este comando registrará el equipo como cliente RaptorXML Server con Altova LicenseServer. Para aprender a asignar licencias al equipo y a comprender el proceso de asignación de licencias consulte la documentación de Altova LicenseServer.

Implementación en Linux

Para implementar la aplicación **`registerlicense`** con el paquete wheel de Python, las bibliotecas compartidas que aparecen a continuación deben estar presentes en un directorio **`lib`** del mismo nivel. Las bibliotecas compartidas se pueden copiar desde la carpeta de instalación de RaptorXML:

```
/opt/Altova/RaptorXMLServerRaptorXMLServer2024/lib
```

- `libcrypto.so.1.0.0`
- `libssl.so.1.0.0`

- `libstdc++.so.6`
- `libtbb.so.2`

© 2017-2023 Altova GmbH

7.2 API de Python

La API de Python de RaptorXML permite acceder y gestionar datos de documentos XML y esquemas XML mediante scripts de Python. Estos son algunos de los usos típicos de la API de Python:

- implementar reglas de validación y mensajes de error personalizados
- exportar contenido de documentos XML a una BD
- exportar contenido de documentos XML a formatos de datos personalizados
- explorar y recuperar datos de forma interactiva del modelo de datos de documentos XML desde una shell de Python o un notebook de Jupyter (https://jupyter.org/)

Interfaces API de Python

Las API de Python (para XML y XSD) ofrecen acceso a metadatos, información estructural y datos incluidos en documentos XML y XSD. Como resultado, puede crear scripts Python que utilicen las API para acceder y procesar los datos de estos documentos. Por ejemplo, se puede enviar un script Python a RaptorXML Server que escriba datos desde un documento XML en una base de datos o en un archivo CSV.

Puede encontrar ejemplos de scripts para las API de Python de RaptorXML en https://github.com/altova

También puede consultar la referencia de las API de Python en el sitio web de Altova:

- Referencia de la API de Python (versión 1)
- Referencia de la API de Python (versión 2)

Nota: La **versión 1 de la API de Python de Raptor está obsoleta**. Use en su lugar la versión 2 de la API de Python.

Paquete de RaptorXML Server para Python

En la instalación de RaptorXML Server también encontrará un paquete Python en formato wheel. Puede usar el comando `pip` de Python para instalar este paquete como módulo de la instalación Python. Tras instalar el módulo RaptorXML, podrá usar sus funciones dentro del código. De este modo, las características y funciones de RaptorXML se pueden usar fácilmente en cualquier programa Python junto con bibliotecas Python de otros autores, como bibliotecas gráficas, por ejemplo.

Para más información sobre cómo utilizar el paquete de RaptorXML Server para Python consulte la sección RaptorXML Server como paquete Python 374.

Nota: La versión 2024 y versiones posteriores del archivo wheel de Python son compatibles con la versión 3.11.5 y versiones posteriores de Python.

Scripts Python

Los scripts Python creados por el usuario se pueden enviar a RaptorXML con el parámetro `--script` de estos comandos:

- `valxml-withxsd (xsi)` 63
- `valxsd (xsd)` 75

Estos comandos que invocan a scripts Python se pueden usar tanto en la interfaz de la línea de comandos (ILC) 58 como en la interfaz HTTP 254. El uso de scripts Python dentro de las API de Python de RaptorXML Server se documenta en https://github.com/altova.

Trabajar con scripts Python seguros

Cuando se especifica un script Python por HTTP para RaptorXML Server, el script solo funciona si está ubicado en el directorio de confianza 259. El script se ejecuta desde el directorio de confianza. Si especifica un script de cualquier otro directorio, se produce un error. El directorio de confianza se define en la opción `server.script-root-dir` 259 del archivo de configuración del servidor 258 y **es obligatorio** especificar un directorio de confianza si quiere usar scripts Python. Por tanto, asegúrese de guardar en este directorio todos los scripts Python que desea usar.

Aunque todos los resultados generados por el servidor para solicitudes de trabajo HTTP se escriben en el directorio de salida de trabajos 259 (que es un subdirectorio de `output-root-directory` 259), esta limitación no afecta a los scripts Python, que pueden escribir en cualquier ubicación. El administrador del servidor debería revisar los scripts Python del directorio de confianza 259 para evitar problemas de seguridad.

7.2.1 Versiones de la API de Python

RaptorXML Server ofrece varias versiones de la API de Python. Todas las versiones antiguas de la API de Python son compatibles con la versión actual de RaptorXML Server. La versión de la API de Python se selecciona con la marca de la línea de comandos **`--script-api-version=MAJOR_VERSION`**. El valor predeterminado del argumento `MAJOR_VERSION` siempre es la versión actual. Cada vez que se introduzcan mejoras o cambios incompatibles, estará disponible un `MAJOR_VERSION` nuevo para la API de Python de RaptorXML Server. El usuario de la API no tiene que actualizar sus scripts cuando se publiquen versiones nuevas.

Sin embargo, recomendamos:

- Utilizar la marca `--script-api-version=MAJOR_VERSION` para invocar scripts de otras utilidades desde la línea de comandos de RaptorXML Server (o desde la API web). Así se garantiza que los scripts sigan funcionando tras la instalación de actualizaciones de RaptorXML Server (incluso si se publica una versión nueva de `MAJOR_VERSION`).
- Utilizar la versión más reciente de la API para proyectos nuevos, aunque las versiones nuevas de RaptorXML Server sean compatibles con versiones antiguas de la API.

A continuación se enumeran las versiones de la API de Python que están disponibles actualmente. La documentación de estas API se puede consultar en el sitio web de Altova (ver enlaces más abajo).

Archivos de ejemplo

Puede encontrar ejemplos de scripts para las API de Python de RaptorXML en https://github.com/altova.

API de Python (versión 1)

Se introdujo con RaptorXML Server v2014.

Marca de la línea de comandos: **`--script-api-version=1`**

 © 2017-2023 Altova GmbH

Documentación: Referencia de la API de Python (versión 1)

Se trata de la API de Python de RaptorXML Server original. Ofrece funciones para acceder al modelo interno de RaptorXML Server para:

- XML 1.0 y XML 1.1 (módulo API **`xml`**)
- XMLSchema 1.0 y XMLSchema 1.1 (módulo API **`xsd`**)
- XBRL 2.1 (módulo API **`xbrl`**)

La API se puede utilizar a través de varias funciones de devolución de llamada que se implementan en un archivo de script Python.

- `on_xsi_valid`
- `on_xsd_valid`
- `on_dts_valid`
- `on_xbrl_valid`

El script se especifica con la opción `--script` en la línea de comandos. Las funciones de devolución de llamada se invocan solamente si la validación finaliza correctamente. Para obtener más información sobre las funciones de devolución de llamada y la API, consulte la Referencia de la API de Python (versión 1) en el sitio web de Altova.

Nota: La **versión 1 de la API de Python de Raptor está obsoleta**. Use en su lugar la versión 2 de la API de Python.

API de Python (versión 2)

Se introdujo con RaptorXML Server v2015r3. La versión más reciente de la API es la versión **`2.8.6`**.

Marca de la línea de comandos	Edición
`--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*
Documentación: Referencia de la API de Python (versión 2)	

En esta versión las funciones de devolución de llamada no solamente se invocan cuando la validación finaliza correctamente, sino también cuando se produce un error durante la validación. Por ello se ha modificado el nombre de las funciones de devolución de llamada:

- `on_xsi_finished`
- `on_xsd_finished`
- `on_dts_finished`
- `on_xbrl_finished`

Además, RaptorXML Server ofrece varias opciones `--script`. Las devoluciones de llamada implementadas en los scripts Python se ejecutan en el orden indicado en la línea de comandos.

7.2.2 RaptorXML Server como paquete Python

A partir de la versión 2024 de RaptorXML Server, la API de Python está disponible como paquete wheel nativo de Python para **Python 3.11.5**. El paquete wheel de Python puede instalarse como módulo de extensión en la distribución de Python 3.5 que usted prefiere (p. ej. la distribución de python.org). Algunas distribuciones de Python 3 (p. ej. las de jupyter.org, anaconda.org y SciPy.org) incluyen una amplia gama de módulos de extensión para datos masivos, matemáticas, ciencia, ingeniería y gráficos. Estos módulos ya pueden estar a disposición de RaptorXML Server sin necesidad de generarlos para RaptorXML Server específicamente. Por lo demás, el paquete wheel funciona igual que la aplicación **`RaptorXMLXBRL-python.exe`** que viene con RaptorXML Server.

Nota: el paquete wheel de Python es un módulo de extensión de Python 3.11.5 y debe coincidir con la versión 3.11.5 de Python.

Nota: el paquete wheel de Python no incluye la API de Python v1.

Nota: si actualiza su versión de RaptorXML Server, asegúrese de que actualiza el paquete wheel de Python en su entorno Python.

En este apartado encontrará toda la información necesaria para instalar el paquete RaptorXML Server correctamente:

- Nombre del archivo wheel 375
- Ubicación del archivo wheel 375
- Instalar un archivo wheel con pip 375
- Solución de problemas en la instalación 375
- El archivo de catálogo raíz 376
- El archivo de configuración JSON 376

 © 2017-2023 Altova GmbH

Para más información sobre cómo usar la API de Python de RaptorXML Server consulte la referencia de la API de Python y los ejemplos[372]. También encontrará ejemplos de scripts que usan la API de Python en https://github.com/altova.

Nombre del archivo wheel

El nombre de los archivos wheel siguen este patrón:

raptorxmlserver-{versión}(-{marca compilación})?-{marca python}-{marca abi}-{marca plataforma}.whl

Ejemplo:
raptorxmlserver-2.8.1-cp35-cp35m-win_amd64.whl

Ubicación del archivo wheel

La instalación de RaptorXML Server viene con un archivo wheel, ubicado en la carpeta **bin** de la aplicación:

Windows	`Archivos de programa\Altova\RaptorXMLServer2024\bin`
Linux	`/opt/Altova/RaptorXMLServer2024/bin`
Mac	`/usr/local/Altova/RaptorXMLServer2024/bin`

Instalar un archivo wheel con pip

Utilice el comando **pip** para instalar el paquete RaptorXML Server como módulo de Python:

```
pip install <archivo-wheel>.whl
python -m pip install <archivo-wheel>.whl
```

Si instaló Python 3.11.5 o superior desde python.org, entonces también tendrá instalado `pip`. Si no es así, primero deberá instalar **pip**. Visite https://docs.python.org/3/installing/ para obtener más información.

Solución de problemas en la instalación

Si está usando una versión antigua del intérprete de Python, puede que necesite realizar algunas modificaciones durante la instalación para usar las bibliotecas **vcruntime** más recientes o las bibliotecas C++ estándar en Unix. Estas bibliotecas vienen incluidas en el paquete de instalación de RaptorXML Server y se pueden usar como explicamos a continuación.

Windows
Si falta el archivo **vcruntime140_1.dll**, cópielo desde la carpeta `Program Files\Altova\RaptorXMLServer2024\bin` a la carpeta de instalación de Python (la carpeta que contiene **python.exe**). (Lo importante es que el intérprete de Python debe saber dónde encontrar los archivos DLL o las bibliotecas compartidas.)

Linux
Si la biblioteca C++ de su sistema está obsoleta, el intérprete de Python no sabrá cómo encontrar la biblioteca C++ más recuente que usa el paquete de Python que viene con RaptorXML Server. Para solucionar este problema, **$LD_LIBRARY_PATH** debe apuntar a la biblioteca nueva de la carpeta de RaptorXML Server: `$ export LD_LIBRARY_PATH=/opt/Altova/RaptorXMLServer2024/lib`.

macOS
Si la biblioteca C++ de su sistema está obsoleta, el intérprete de Python no sabrá cómo encontrar la biblioteca C++ más recuente que usa el paquete de Python que viene con RaptorXML Server. Para solucionar este problema, **$DYLD_LIBRARY_PATH** debe apuntar a la biblioteca nueva de la carpeta de RaptorXML Server: `$ export DYLD_LIBRARY_PATH=/usr/local/Altova/RaptorXMLServer2024/lib`.

El archivo de catálogo raíz

El módulo RaptorXML para Python debe ser capaz de encontrar **RootCatalog.xml**, el archivo de catálogo raíz que está en la carpeta de instalación de RaptorXML Server. El módulo RaptorXML necesita el catálogo para encontrar los diferentes recursos (como esquemas y especificaciones) a los que hace referencia el módulo para llevar a cabo diferentes funciones, como validaciones y transformaciones, por ejemplo. El módulo RaptorXML encontrará **RootCatalog.xml** automáticamente si la ubicación del catálogo no cambió desde que se instaló RaptorXML Server.

En caso de que se modificara el entorno de RaptorXML Server o de que se moviera **RootCatalog.xml** a otra ubicación, podrá especificar la ubicación del catálogo mediante variables de entorno y desde el archivo de configuración JSON del módulo RaptorXML 374. En la lista que aparece a continuación puede ver varios métodos diferentes. El módulo RaptorXML determina la ubicación de **RootCatalog.xml** buscando estos recursos en el orden especificado en la tabla.

1	Variable de entorno `ALTOVA_RAPTORXML_PYTHON_CATALOGPATH`	Crear con un valor que sea la ruta de acceso de **RootCatalog.xml**
2	Registro HKLM: `SOFTWARE\Altova\RaptorXMLServer\Installation_v2024_x64\Setup\CatalogPath`	La clave de registro la añade el programa de instalación de RaptorXML Server. Su valor es la ruta de acceso de **RootCatalog.xml**. (*Solo para Windows.*)
3	Ubicación: `/opt/Altova/RaptorXMLServer2024/etc/RootCatalog.xml`	*Sólo para Linux.*
4	Ubicación: `/usr/local/Altova/RaptorXMLServer2024/etc/RootCatalog.xml`	*Sólo para Mac.*
5	Variable de entorno `ALTOVA_RAPTORXML_PYTHON_CONFIG`	Crear con un valor que sea la ruta de acceso del archivo de configuración JSON 376.
6	Ubicación: `.altova/`**raptorxml-python.config**	El archivo de configuración JSON 376 que está en el directorio de trabajo actual.
7	Ubicación: `~/.config/altova/`**raptorxml-python.config**	El archivo de configuración JSON 376 que está en el directorio de inicio del usuario.
8	Ubicación: `/etc/altova/altova/`**raptorxml-python.config**	El archivo de configuración JSON 376. (*Solo para Linux y Mac.*)

El archivo de configuración JSON

Puede crear una archivo de configuración JSON para el módulo RaptorXMLServer. Este archivo se usará en las opciones 5, 6, 7 y 8 de la tabla anterior para buscar el archivo de catálogo raíz 376. El archivo de configuración

© 2017-2023 Altova GmbH

JSON debe contener un mapa con una clave `"CatalogPath"` cuyo valor sea la ruta de acceso del archivo de catálogo raíz 376.

Fragmento de un archivo de configuración JSON

```
{
    "CatalogPath": "/path/to/RootCatalog.xml"
}
```

7.2.3 Depurar scripts de Python del lado servidor

La mayoría de las funciones de depuración, aparte de las devoluciones de llamada específicas del servidor, se pueden usar en un intérprete Python estándar o en un entorno (virtual) Python después de haber instalado el módulo de RaptorXML Server con el comando `pip`:

```
pip install –upgrade "/path/to/RaptorXML/application-folder/bin/raptorxml-version-cp37-cp37m-winversion.whl"
```

Después de instalar la rueda debería poder usar cualquier IDE de Python para depurar un script. Puede intentar extraer la función principal como una función aparte que toma un objeto de instancia. Después puede llamar a esta función (i) con las devoluciones de llamada de RaptorXML o (ii) ejecutando directamente el script con un intérprete Python.

```
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 Depurar scripts de Python en Visual Studio Code

En este apartado asumimos que tiene instalada una versión actual de Visual Studio Code (VS Code) con la extensión **ms-python.python**. Lea con detenimiento la guía oficial de configuraciones de Python en Visual Studio Code.

Tenga en cuenta que:

- Esta guía usa **raptorxml-python** como comando para ejecutar RaptorXML Server como intérprete de Python.
- El ejecutable **raptorxml-python** está disponible en la carpeta **bin** de su carpeta de aplicación de RaptorXML Server.

Resumen

A continuación explicamos dos métodos para usar VS Code para depurar scripts de Python en RaptorXML Server.

- El método 1 también funciona con servidores y devoluciones de llamada Python con RaptorXML (**--script** option).
- El método 2 no necesita que modifique el código fuente. Se trata de una invocación modificada de RaptorXML. Este método no funciona con servidores ni devoluciones de llamada Python con RaptorXML (**--script** option).
- Los dos métodos funcionan con intérpretes Python estándar y el módulo Python de RaptorXML importado (**'import altova_api.v2 as altova'**).

Método 1: cambiar el código fuente

Siga estos pasos:

1. Ejecute: **raptorxml-python -m pip install --upgrade debugpy**
2. Añada estas líneas a su código fuente Python:

```
python
import debugpy
debugpy.listen(5678)
debugpy.wait_for_client()
debugpy.breakpoint()
```

3. Copie esta configuración de lanzamiento en el archivo **launch.json** de VS Code (los valores predeterminados sirven para los valores anteriores) y selecciónela para ejecutar el comando **Run**.

```
json5
  {
    "name": "Python: Remote Attach",
    "type": "python",
    "request": "attach",
    "connect": {
    "host": "localhost",
    "port": 5678
    },
    "pathMappings": [
```

 © 2017-2023 Altova GmbH

```
        {
          "localRoot": "${workspaceFolder}",
          "remoteRoot": "."
        }
      ]
    }
```

También puede llevar a cabo la ejecución con el comando de menú **Run->Add Configuration...->Python->Remote Attach** con los valores predeterminados aceptados.

4. Ejecute su script de Python (o RaptorXML con devoluciones de llamada `--script`) normalmente.
5. Empiece a depurar (por lo general con la tecla de acceso rápido **F5**).

Método 2: usar una línea de comandos modificada

Siga estos pasos:

1. Añada una configuración de lanzamiento (como en el método 1) y selecciónela para ejecutar el comando `Run`.
2. Añada un punto de interrupción a su script de Python.
3. Ejecute el comando: `raptorxml-python -m debugpy --listen 0.0.0.0:5678 --wait-for-client your-script-.py`
4. Empiece a depurar (por lo general con la tecla de acceso rápido **F5**).

Nota: la depuración también funciona con contenedores y servidores remotos. Tiene que cambiar la clave `host` de la entrada `connect` en la configuración de lanzamiento. También puede usar otros puertos, siempre que el código o la línea de comandos y `launch.json` tengan los mismos valores.

Configurar raptorxml-python.exe como intérprete predeterminado para VS Code

Puede configurar `raptorxml-python.exe` como intérprete Python predeterminado para VS Code. Para ello añada este fragmento de código a su archivo `settigs.json` de VS Code:

```
json
  "python.defaultInterpreterPath": "/path/to/raptorxml-python.exe"
   ...
```

En este caso también es posible usar una configuración de lanzamiento "Archivo actual" que inicie el script de depuración. Consulte la documentación oficial de VS Code para más detalles.

7.2.5 Preguntas frecuentes

P: *Quiero escribir un script de Python que cree una instancia XML nueva, un elemento después de otro, mientras se ejecuta dentro del servidor raptor. Estas instancias tienen que estar serializadas para los resultados con distintos cifrados y formatos según los parámetros. ¿Se puede hacer esto con RaptorXML Server?*

R: No, de momento no es posible porque no contamos con una API para crear instancias XML arbitrarias. Sin embargo, a la hora de generar instancias XBRL sí que tenemos una API de alto nivel que administra muchos de los detalles técnicos (como evitar escribir duplicados de contextos o unidades, etc.). Consulte

https://www.altova.com/manual/en/raptorapi/pyapiv2/2.8.6/html/xbrl.InstanceDocumentBuilder.html para más información.

P: *Quiero usar lxml. ¿Puedo instalar bibliotecas lxml en la carpeta Python en "RaptorXMLXBRLServer2024/lib/"?*

R: Puede instalar la mayoría de módulos Python ejecutando este comando directamente en una terminal con derechos de administrador:

```
"/path/to/RaptorXML/application-folder/bin/RaptorXMLXBRL-python.exe" -m pip install lxml
```

P: *¿Podría crear una cadena grande que contenga la instancia XML, analizarla y volver a serializarla?*

R: Es una posibilidad. Puede analizar y validas instancias XML y XBRL desde un búfer de cadena con la API de Python:

```
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())
```


© 2017-2023 Altova GmbH

7.3 API de .NET Framework

La **API de .NET Framework** de RaptorXML Server permite integrar el motor de RaptorXML Server en aplicaciones escritas en C# y en otros lenguajes .NET.

Se implementa como ensamblado .NET y coloca el motor de RaptorXML Server dentro de una aplicación directamente o de un mecanismo de extensión basado en .NET Framework como VSTO (Visual Studio Tools for Office). La API ofrece acceso a funciones de validación de documentos y consulta del modelo de datos desde RaptorXML Server. El mecanismo utilizado es similar al que se utiliza para la API de Python de RaptorXML Server.

Referencia y recursos

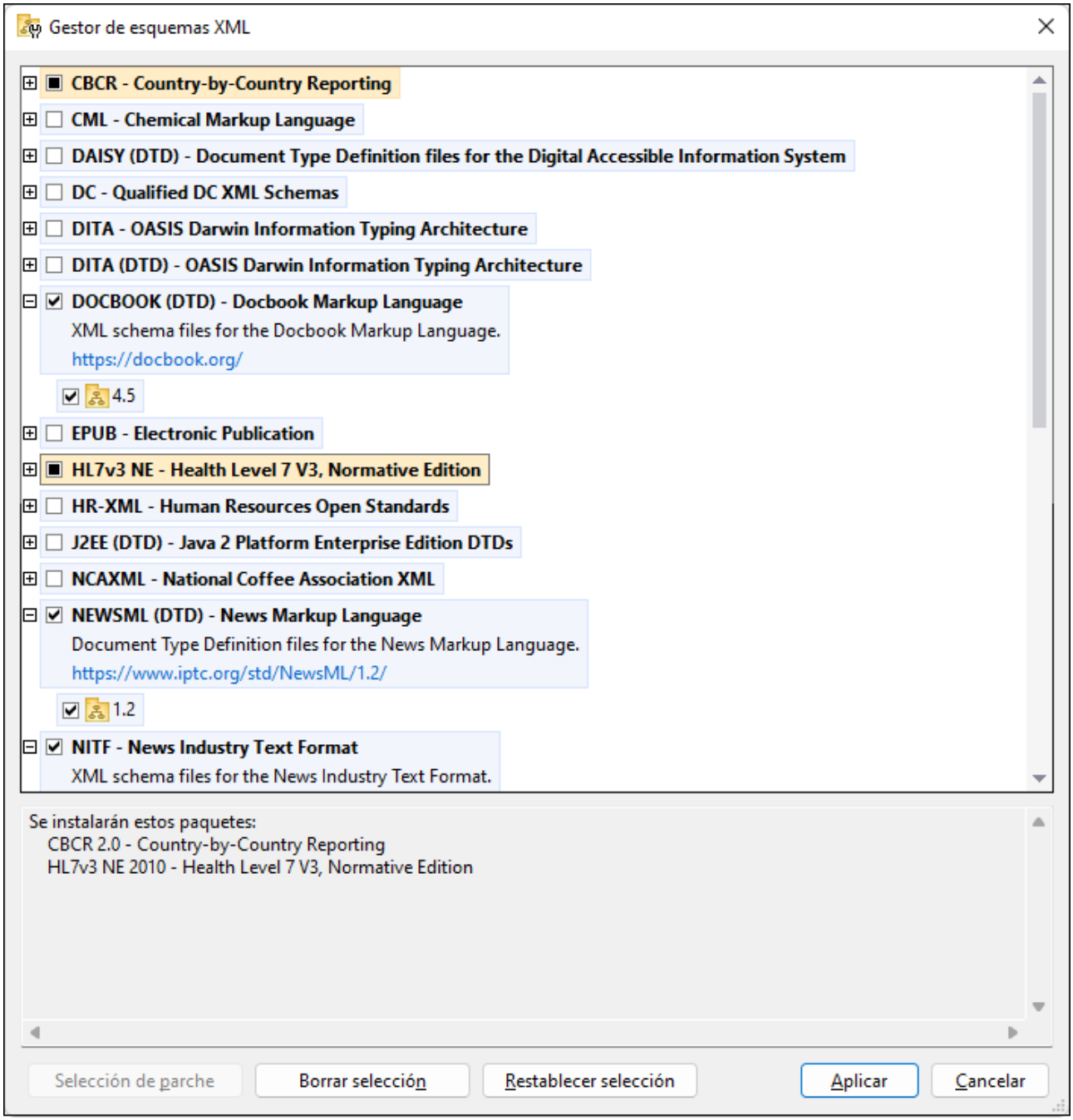
- *Documentación de la API:* la versión más reciente de la documentación de la API de .NET Framework de RaptorXML Server puede consultarse aquí: https://www.altova.com/manual/en/raptorapi/dotnetapiv2/2.8.6/html/index.html.
- *Ejemplo de código:* puede consultar un ejemplo en https://github.com/altova/RaptorXML-Examples.

8 Gestor de esquemas

El Gestor de esquemas XML es una herramienta que ofrece una forma centralizada de instalar y administrar esquemas XML (DTDs para XML y esquemas XML) para usarlos en todas las aplicaciones de Altova compatibles con XML Schema, incluido RaptorXML Server.

- En Windows, el gestor tiene una interfaz gráfica del usuario (*imagen siguiente*) a la que también puede acceder desde la línea de comandos. (Las aplicaciones de escritorio de Altova solo están disponibles para Windows; *consulte la lista siguiente*.)
- En Linux y macOS el Gestor de esquemas solo está disponible en la línea de comandos. (Las aplicaciones de escritorio de Altova están disponibles para Windows, Linux y macOS; *consulte la lista siguiente*.)

© 2017-2023 Altova GmbH



Aplicaciones de Altova que funcionan con el Gestor de esquemas

Aplicaciones de escritorio (solo para Windows)	Aplicaciones de servidor (Windows, Linux, macOS)
XMLSpy (todas las ediciones)	RaptorXML Server, RaptorXML+XBRL Server
MapForce (todas las ediciones)	StyleVision Server

StyleVision (todas las ediciones)	

Instalación y desinstalación del Gestor de esquemas

El Gestor de esquemas se instala automáticamente al instalar cualquiera de las aplicaciones de Altova compatibles con XML o el Altova Mission Kit (*véase la tabla de más arriba*).

También se elimina automáticamente si desinstala todas las aplicaciones de Altova compatibles con XML del equipo.

Características de Gestor de esquemas

El Gestor de esquemas permite:

- Ver los esquemas XML que hay instaladas en su equipo y comprobar si hay versiones nuevas para descargar.
- Descargar las versiones más recientes de los esquemas XML independientemente del ciclo de versiones de Altova. Altova guarda todos los esquemas en un sistema de almacenamiento en línea al que tiene acceso el Gestor de esquemas y desde donde puede descargarlas tan pronto como estén disponibles.
- Instalar o desinstalas cualquiera de las múltiples versiones de un esquema en concreto (o todas ellas, si las necesita).
- Un solo esquema XML representa un "paquete", pero puede tener dependencias en otros esquemas. Al instalar o desinstalar un esquema, se detectan e instalan o desinstalan también automáticamente todas sus dependencias. La interfaz gráfica del usuario (o la línea de comandos, en su caso) le informa cuando se añaden o eliminan esquemas.
- Los esquemas XML administradas con el Gestor de esquemas pueden usar el catálogo XML, que permite resolver referencias a URI en documentos de instancia o esquema desde archivos locales, en vez de a través de Internet.
- Todos los esquemas principales están incluidos en Gestor de esquemas y se actualizan de forma periódica a la versión más reciente. De esta forma puede administrar todos los esquemas desde un punto común y tenerlos siempre listos para las aplicaciones de Altova que las usan.
- Los cambios que se realizan en el Gestor de esquemas afectan a todos los productos de Altova que estén instalados en ese equipo.
- En los productos de Altova, si intenta validar con un esquema que no está instalado pero sí disponible con el Gestor de esquemas, este se instala automáticamente. Sin embargo, si el paquete de esquemas que quiere instalar contiene asignaciones de espacios de nombres, no puede instalarse automáticamente, sino que debe ejecutar Gestor de esquemas, seleccionar qué paquetes quiere instalar y ejecutar la instalación. Si después de instalar los paquetes la aplicación de Altova que está abierta no se reinicia automáticamente, debe reiniciarla manualmente.

Funcionamiento

Altova mantiene un almacenamiento en línea donde guarda todos los esquemas XML de los productos de Altova. Este almacenamiento se actualiza de forma periódica, por ejemplo, poco después de que las organizaciones correspondientes publiquen las versiones nuevas de los esquemas respectivos. Al ejecutar Gestor de esquemas desde la interfaz gráfica del usuario aparece información sobre los esquemas más recientes disponibles en un cuadro de diálogo en el que puede visualizarlos, instalarlos, actualizarlos o desinstalarlos.

También puede instalar los esquemas de otra manera. En el sitio web de Altova (https://www.altova.com/schema-manager) puede seleccionar el esquema y los esquemas dependientes de

© 2017-2023 Altova GmbH

este que quiere instalar. El sitio web prepara un archivo de tipo `.altova_xmlschemas` que puede descargar y que contiene la información sobre los esquemas seleccionados. Al hacer doble clic en este archivo o pasarlo a Gestor de esquemas desde la línea de comandos como argumento del comando **install** 396, Gestor de esquemas instala los esquemas que contiene.

Memoria caché local: seguimiento de esquemas

Independientemente de cómo se instalen los esquemas, toda la información sobre los esquemas instalados se almacena en una ubicación centralizada de su equipo, el directorio caché. El directorio caché local está en:

Windows	`C:\ProgramData\Altova\pkgs\.cache`
Linux	`/var/opt/Altova/pkgs\.cache`
macOS	`/var/Altova/pkgs`

El directorio caché local se actualiza automáticamente de vez en cuando para que el estado más actual del equipo corresponda con el del almacenamiento en línea. Más concretamente, el caché se actualiza:

- al ejecutar el Gestor de esquemas.
- al ejecutar RaptorXML Server por primera vez en un mismo día natural.
- si RaptorXML Server ya se está ejecutando, el directorio caché se actualiza cada 24 horas.
- también puede actualizar el caché local desde el almacenamiento en línea manualmente ejecutando el comando de actualización **update** 399 desde la línea de comandos.

Si instala o desinstala esquemas, el directorio caché local se actualiza automáticamente con información sobre los esquemas disponibles e instalados, además de con los propios archivos de esquema.

No modifique la memoria caché manualmente

El directorio caché local se mantiene automáticamente en base a los esquemas que instale o desinstale; no debe modificarlo ni eliminarlo manualmente. Si necesita restaurar el Gestor de esquemas a su estado original, ejecute el comando **reset** 397 desde la línea de comandos y después ejecute el comando **initialize** 395.

8.1 Ejecutar el gestor de esquemas

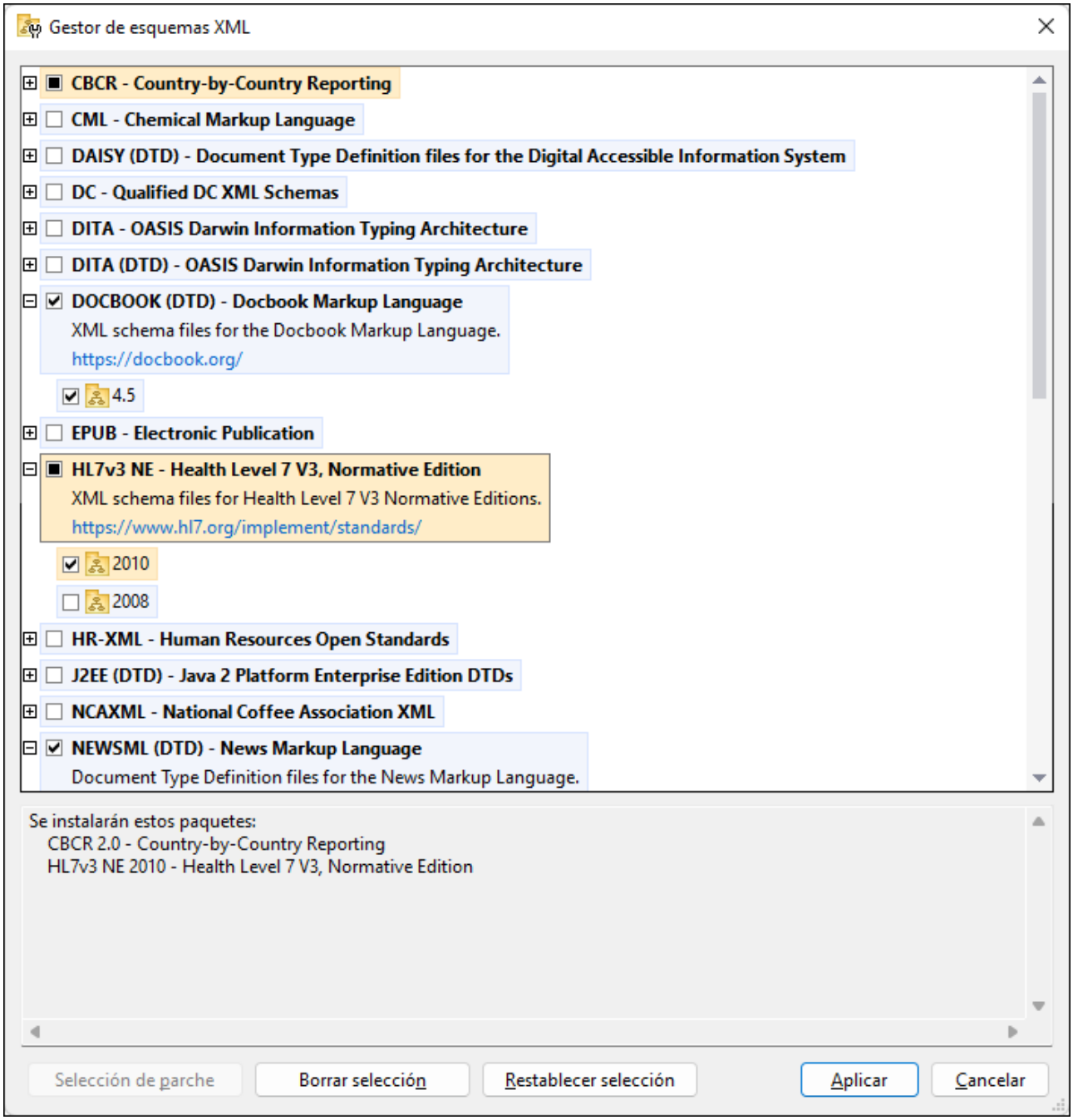
Interfaz gráfica del usuario

Hay varias formas de acceder a la IGU del Gestor de esquemas:

- *Durante la instalación de RaptorXML Server:* al final del proceso de instalación, seleccione la casilla *Invocar al Gestor de esquemas* para acceder directamente a la IGU del gestor de esquemas XML. Con él puede instalar esquemas durante el proceso de instalación de su aplicación de Altova.
- *Después de la instalación de RaptorXML Server:* una vez haya instalado la aplicación puede acceder al Gestor de esquemas en cualquier momento desde el comando de menú Herramientas | **Gestor de esquemas XML**.
- Mediante el archivo `.altova_schemas` que descargó del *sitio web de Altova:* haga doble clic en el archivo para ejecutar Gestor de esquemas, que instalará los esquemas que haya seleccionado.

Cuando se abra la IGU del Gestor de esquemas (*imagen siguiente*) podrá ver en ella los esquemas que ya se han instalado. Si quiere instalar más solo tiene que seleccionarlas, y al contrario si quiere desinstalar alguna. Una vez haya terminado, puede aplicar los cambios. los esquemas que se vayan a instalar o desinstalar aparecerán resaltados y un mensaje le avisará de los cambios que está a punto de hacer en la ventana *Mensajes*, en la parte inferior de la ventana de Gestor de esquemas (*véase imagen*).

© 2017-2023 Altova GmbH



Interfaz de la línea de comandos

Para ejecutar el Gestor de esquemas desde una interfaz de la línea de comandos debe usar su archivo ejecutable, **`xmlschemamanager.exe`**.

Puede encontrar este archivo:

- *en Windows:* `C:\ProgramData\Altova\SharedBetweenVersions`
- *en Linux o macOS (solo para aplicaciones de servidor):* `%INSTALLDIR%/bin`, donde `%INSTALLDIR%` es el directorio de instalación del programa.

Puede usar cualquiera de los comandos de la referencia de la línea de comandos 394, a continuación.

Para mostrar al ayuda de los comandos ejecute:

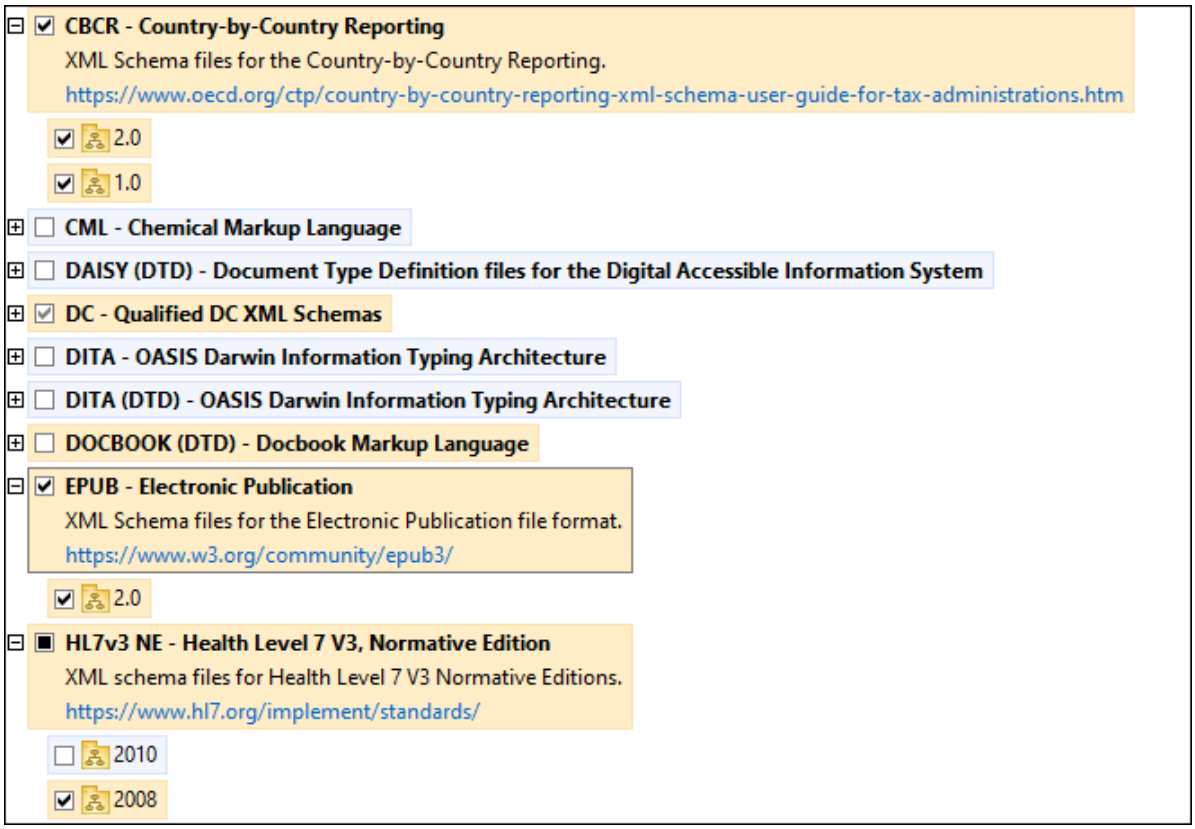
- *en Windows:* `xmlschemamanager.exe --help`
- *en Linux o macOS (solo para aplicaciones de servidor):* `sudo ./xmlschemamanager --help`

 © 2017-2023 Altova GmbH

8.2 Categorías de estado

Gestor de esquemas diferencia los esquemas que administra entre:

- *Esquemas instalados:* estos aparecen en la IGU con sus casillas marcadas (*en la imagen siguiente las versiones marcadas de los esquemas EPUB y HL7v3 NE son las que están instaladas*). Si se seleccionan todas las versiones de un esquema, en la casilla del esquema aparece una marca de verificación. Si hay al menos un esquema sin seleccionar, en la en la casilla del esquema aparece un cuadrado negro. Para **desinstalar** un esquema debe desmarcar la casilla correspondiente (*en la imagen siguiente, el DTD DocBook está instalado y su casilla se ha desmarcado, es decir, se va a desinstalar*).
- *Esquemas disponibles no instalados:* estos aparecen en la IGU con las casillas correspondientes sin seleccionar. Para **instalar** esquemas, marque la casilla correspondiente.



- *Esquemas que pueden actualizarse:* son los que han sido revisados por sus emisores. Aparecen indicados en la IGU con el icono (*imagen anterior*). Puede aplicar **parches** al esquema seleccionado con la revisión que esté disponible.

Puntos importantes

- En la imagen anterior se han marcado los esquemas CBCR. Las que tienen un fondo azul ya están instaladas. Las que tienen el fondo amarillo no están instaladas pero se han seleccionado para instalarlas. Observe que el esquema HL7v3 NE 2010 no está instalada ni se ha seleccionado para instalarlo.

- Al ejecutar el Gestor de esquemas desde la línea de comandos puede usar el comando **list** con distintas opciones para ver distintas categorías de esquemas:

xmlschemamanager.exe list	Muestra todas los esquemas instalados y disponibles; también indica qué esquemas se pueden actualizar
xmlschemamanager.exe list -i	Muestra solo los esquemas instalados; también indica qué esquemas se pueden actualizar
xmlschemamanager.exe list -u	Muestra qué esquemas se pueden actualizar

Nota: en Linux y macOS **use sudo ./xmlschemamanager list**

© 2017-2023 Altova GmbH

8.3 Aplicar parches o instalar un esquema

Aplicar un parche a un esquema instalado

A veces los emisores de los esquemas XML generan parches. Cuando el Gestor de esquemas XML detecta que hay parches disponibles, estos aparecen en las listas de esquemas, desde donde puede instalarlos.

En la IGU

Los parches se indican con el icono . (*Consulte también el apartado anterior sobre categorías de esquemas* 389.) Si hay parches disponibles se habilita el botón **Seleccionar parches**. Haga clic en él para seleccionar y preparar los parches. En la IGU, el icono de los esquemas correspondientes cambia de a y el cuadro de diálogo le informa de qué parches se van a aplicar. Las listas del panel principal y las del panel Mensajes están ordenadas alfabéticamente, por lo que puede ver y revisar los esquemas antes de aplicar los parches. Una vez esté listo para instalar los parches seleccionados, haga clic en **Aplicar**.

En la línea de comandos

Para aplicar un parche desde lal ínea de comandos:

1. Ejecute el comando `list -u` 396. Aparece una lista con los esquemas para las que hay parches disponibles.
1. Ejecute el comando `upgrade` 399 para instalar todos los parches.

Instalar un esquema disponible

Para instalar esquemas puede usar la IGU del Gestor de esquemas o enviar las instrucciones al Gestor de esquemas desde la línea de comandos.

Nota: si el esquema actual tiene dependencias en otros esquemas, también se instalan (o desinstalan, según el caso) los esquemas dependientes.

En la IGU

Para instalar esquemas con la IGU del Gestor de esquemas, seleccione los esquemas que quiere instalar y haga clic en **Aplicar**.

También puede seleccionar los esquemas que quiere instalar en el sitio web de Altova y generar desde allí un archivo `.altova_schemas`. Al hacer doble clic en este archivo se abre el Gestor de esquemas con los esquemas que indicó preseleccionados. Solo tiene que hacer clic en **Aplicar**.

En la línea de comandos

Para instalar esquemas desde la línea de comandos ejecute el comando `install`:

```
xmlschemamanager.exe install [opciones] Schema+
```

donde `FILTER` es el esquema (o los esquemas) que quiere instalar o un archivo `.altova_schemas`. Para hacer referencia a un esquema se usa un identificados con el formato `<nombre>-<versión>` que aparece junto a cada esquema que muestra el comando `list` 396. Puede introducir tantos esquemas como quiera. Para más detalles consulte la descripción del comando `install` 396.

Nota: en Linux o macOS, use el comando `sudo ./xmlschemamanager`.

Instalar un esquema requerido
Si ejecuta un comando en RaptorXML Server y RaptorXML Server descubre que uno de los esquemas que necesita para ejecutar el comando falta o está incompleta, el Gestor de esquemas incluirá información sobre ese componente de esquema que falta. Entonces puede aplicar el parche indicado y/o instalar el esquema que falta.

Siempre puede ver todos los esquemas instalados previamente ejecutando el Gestor de esquemas desde **Herramientas | Gestor de esquemas**.

© 2017-2023 Altova GmbH

8.4 Desinstalar o restaurar esquemas

Desinstalar un esquema

Para desinstalar esquemas puede usar la IGU del Gestor de esquemas o enviar las instrucciones al Gestor de esquemas desde la línea de comandos.

Nota: si el esquema actual tiene dependencias en otros esquemas, también se instalan (o desinstalan, según el caso) los esquemas dependientes.

En la IGU

Para desinstalar esquemas con la IGU del Gestor de esquemas, seleccione los esquemas que quiere desinstalar y haga clic en **Aplicar**. Los esquemas seleccionadas y sus dependencias se desinstalarán.

Para desinstalar todos los esquemas haga clic en **Deseleccionar todas** y haga clic en **Aplicar**.

En la línea de comandos

Para desinstalar esquemas desde la línea de comandos ejecute el comando `uninstall`[398]:

```
xmlschemamanager.exe uninstall [options] FILTER+
```

donde **FILTER** es el esquema (o los esquemas) que quiere desinstalar o un archivo **.altova_schemas**. Para hacer referencia a un esquema se usa un identificador con el formato **<nombre>-<versión>** que aparece junto a cada esquema que muestra el comando `list`[396]. Puede introducir tantos esquemas como quiera. Para más detalles consulte la descripción del comando `uninstall`[398].

Nota: en Linux o macOS, use el comando **sudo ./xmlschemamanager**.

Restaurar el Gestor de esquemas

Puede restaurar el Gestor de esquemas, es decir, eliminar todos los esquemas instaladas, así como el directorio caché.

- En la IGU, haga clic en **Restaurar selección**.
- En la línea de comandos, use el comando `reset`[397].

Una vez haya ejecutado este comando, asegúrese de que ejecuta también el comando `initialize`[395] para recrear el directorio caché. También puede ejecutar el comando `reset`[397] con la opción `-i`.

Recuerde que `reset -i`[397] restaura la instalación original del producto, por lo que es recomendable ejecutar el comando `update`[399] después de restaurar el gestor. Puede ejecutar el comando `reset`[397] con las opciones `-i` o `-u`.

8.5 Interfaz de la línea de comandos (ILC)

Para llamar a Gestor de esquemas desde la línea de comandos necesita saber la ruta del ejecutable. Por defecto, el ejecutable del Gestor de esquemas se encuentra en:

```
C:\ProgramData\Altova\SharedBetweenVersions\XMLSchemaManager.exe
```

Nota: en los sistemas Linux y macOS una vez haya cambiado el directorio al que contiene el ejecutable, puede llamar al ejecutable con **`sudo ./xmlschemamanager`**.El prefijo **`./`** indica que el ejecutable está en el directorio actual. El prefijo **`sudo`** indica que el comando se debe ejecutar con derechos de administrador.

Sintaxis de la línea de comandos

La sintaxis general para usar la línea de comandos es:

```
<exec> -h | --help | --version | <command> [opciones] [argumentos]
```

En el código anterior la barra vertical **`|`** separa elementos que se excluyen mutuamente. Los corchetes **`[]`** indican elementos opcionales. Básicamente, puede teclear la ruta del ejecutable seguida por las opciones `-h`, `--help` o `--version`, o por un comando. Cada comando puede tener opciones y argumentos. Los comandos se describen en los apartados siguientes.

8.5.1 help

Este comando ofrece ayuda contextual sobre los comandos del ejecutable del Gestor de esquemas.

Sintaxis

```
<exec> help [command]
```

Donde `[command]` es un argumento opcional que indica cualquier nombre válido de comando.

Tenga en cuenta que:

- Puede invocar la ayuda tecleando un comando seguido por `-h` or `--help`, por ejemplo: **`<exec> list -h`**
- Puede invocar la ayuda general tecleando `-h` o `--help` directamente después del ejecutable, por ejemplo:

Ejemplo

Este comando muestra la ayuda del comando `list`:

```
xmlschemamanager help list
```

 © 2017-2023 Altova GmbH

8.5.2 info

Este comando muestra información detallada sobre cada uno de los esquemas dados como argumento. Esta información incluye el título, la versión, la descripción, el editor y las referencias de las dependencias.

Sintaxis

```
<exec> info [options] Schema+
```

- El argumento **Schema** es el nombre de un esquema o parte del nombre de un esquema. (Para ver el ID de un paquete de esquemas y la información relativa a su estado de instalación use el comando list 396.)
- Use **<exec> info -h** para ver la ayuda sobre este comando en la línea de comandos..

Ejemplo

Este comando muestra información detallada sobre los esquemas **DocBook-DTD** y **NITF**:

```
xmlschemamanager info doc nitf
```

8.5.3 initialize

Este comando inicializa el entorno del Gestor de esquemas y crea un directorio caché donde se guardan todos los esquemas localmente. El Gestor de esquemas se inicializa automáticamente la primera vez que instale una aplicación de Altova compatible con él, por lo que normalmente no es necesario ejecutar este comando. Por lo general solo es necesario ejecutarlo después de haber ejecutado el comando `reset`.

Sintaxis

```
<exec> initialize | init [opciones]
```

Opciones
Estas son las opciones del comando **initialize**:

`--help, --h`	Muestra la ayuda sobre este comando en la línea de comandos.
`--silent, --s`	Muestra solamente los mensajes de error. El valor predeterminado es **false**.
`--verbose, --v`	Muestra información suplementaria durante la ejecución. El valor predeterminado es **false**.

Ejemplo

Este comando inicializa el Gestor de esquemas:

```
xmlschemamanager initialize
```

8.5.4 install

Este comando instala una o más esquemas.

Sintaxis

```
<exec> install [options] Schema+
```

Para indicar varios esquemas, repita el argumento `Schema` tantas veces como sea necesario.

El argumento de `Schema` puede ser:

1. Un identificador de esquema en el formato **<name>-<version>**, por ejemplo: **cbcr-2.10**). Para ver todos los identificadores de esquemas y sus versiones ejecute el comando `list` 396. También puede usar el nombre de el esquema abreviado, si este es único, por ejemplo **docbook**. Si usa una abreviación del nombre se desinstalan todos los esquemas que contengan esa abreviación.
2. La ruta de acceso a un archivo **.altova_schemas** descargado desde el sitio web de Altova. Para más información sobre estos archivos consulte la *Introducción al Gestor de esquemas: funcionamiento* 382.

Opciones
Estas son las opciones del comando **install**:

`--help, --h`	Muestra la ayuda sobre este comando en la línea de comandos.
`--silent, --s`	Muestra solamente los mensajes de error. El valor predeterminado es **false**.
`--verbose, --v`	Muestra información suplementaria durante la ejecución. El valor predeterminado es **false**.

Ejemplo

Este comando instala el esquema CBCR 2.0 (Country-By-Country Reporting) y el DTD DocBook más reciente:

```
xmlschemamanager install cbcr-2.0 docbook
```

8.5.5 list

Use este comando para ver los esquemas del Gestor de esquemas; tiene varias opciones:

- lista de todos los esquemas disponibles
- lista de esquemas específicos
- lista de los esquemas instalados
- lista de los esquemas que se pueden actualizar.

Sintaxis

```
<exec> list | ls [options] Schema?
```

 © 2017-2023 Altova GmbH

Si no se indica ningún argumento **schema** la lista incluye todos los esquemas. De lo contrario la lista incluye los esquemas indicados en las opciones (*véase el ejemplo de más abajo*). Recuerde que puede usar el argumento **schema** tantas veces como quiera.

Opciones
Estas son las opciones del comando **list**:

`--help, --h`	Muestra la ayuda sobre este comando en la línea de comandos.
`--installed, --i`	Muestra solamente los esquemas instaladas. El valor predeterminado es **false**.
`--upgradeable, --u`	Muestra solamente los esquemas para las que hay disponible una versión más reciente (parches). El valor predeterminado es **false**.

Ejemplos

- Para ver todos los esquemas disponibles ejecute: **xmlschemamanager list**
- Para ver solamente los esquemas instaladas ejecute: **xmlschemamanager list -i**
- Para ver todos los esquemas cuyos nombres contienen "doc" o "nitf" ejecute: **xmlschemamanager list doc nitf**

8.5.6 reset

Este comando elimina todos los esquemas instalados, así como el directorio de caché. Este comando elimina todos los esquemas instalados y su información. Una vez haya ejecutado este comando, asegúrese de que ejecuta el comando initialize 395 para volver a crear el directorio de caché. También puede ejecutar el comando `reset` con la opción `-i`. Tenga en cuenta que `reset -i` restaura la instalación original del producto, por lo que se recomienda ejecutar también el comando update 399 después de una restauración. También puede ejecutar el comando `reset` con las opciones `-i` y `-u`.

Sintaxis

```
<exec> reset [opciones]
```

Opciones
Estas son las opciones del comando **reset**:

`--help, --h`	Muestra la ayuda sobre este comando en la línea de comandos.
`--init, --i`	Inicializa el entorno del Gestor de esquemas XML después de una restauración. El valor predeterminado es **false**.
`--silent, --s`	Muestra solamente los mensajes de error. El valor predeterminado es **false**.
`--update, --u`	Inicializa y actualiza el entorno del Gestor de esquemas XML después de una restauración. El valor predeterminado es **false**.

`--verbose, --v`	Muestra información suplementaria durante la ejecución. El valor predeterminado es **`false`**.

Ejemplos

- Para restaurar el Gestor de esquemas, ejecute: **`xmlschemamanager reset`**
- Para restaurar el Gestor de esquemas e inicializarlo, ejecute: **`xmlschemamanager reset -i`**
- Para restaurar el Gestor de esquemas, inicializarlo y actualizar la lista de esquemas, ejecute: **`xmlschemamanager reset -i -u`**

8.5.7 uninstall

Este comando desinstala una o más esquemas. Por defecto, cualquier esquema a la que haga referencia el esquema actual también se desinstala. Para desinstalar solamente el esquema actual y mantener aquellas a las que se hace referencia, use la opción `--k`.

Sintaxis

```
<exec> uninstall [opciones] Schema+
```

Para indicar varios esquemas, repita `FILTER` tantas veces como sea necesario.

El argumento de **`Schema`** puede ser:

- Un identificador de esquema en el formato **`<name>-<version>`**, por ejemplo: **`eba-2.10`**). Para ver todos los identificadores de esquemas y sus versiones ejecute el comando `list -i` 396. También puede usar el nombre del esquema abreviado, si este es único, por ejemplo **`eba`**. Si usa una abreviación del nombre se desinstalan todos los esquemas que contengan esa abreviación.
- La ruta de acceso a un archivo **`.altova_taxonomies`** descargado desde el sitio web de Altova. Para más información sobre estos archivos consulte la *Introducción al Gestor de esquemas: funcionamiento* 382.

Opciones
Estas son las opciones del comando **`uninstall`**:

`--help, --h`	Muestra la ayuda sobre este comando en la línea de comandos.
`--keep-references, --k`	Si usa esta opción, los esquemas referenciados no se desinstalan. El valor predeterminado es **`false`**.
`--silent, --s`	Muestra solamente los mensajes de error. El valor predeterminado es **`false`**.
`--verbose, --v`	Muestra información suplementaria durante la ejecución. El valor predeterminado es **`false`**.

 © 2017-2023 Altova GmbH

Ejemplo

Este comando desinstala los esquemas CBCR 2.0 y EPUB 2.0:

```
xmlschemamanager uninstall cbcr-2.0 epub-2.0
```

Este comando desinstala el esquema **eba-2.10** pero no los esquemas a los que hace referencia:

```
xmlschemamanager uninstall --k cbcr-2.0
```

8.5.8 update

Este comando consulta la lista de esquemas disponibles en el almacenamiento en línea y actualiza el directorio de caché local. Esta información se actualiza de forma implícita, por lo que no es necesario ejecutar este comando a no ser que haya ejecutado reset 397 e initialize 395.

Sintaxis

```
<exec> update [opciones]
```

Opciones

Estas son las opciones del comando `update`:

`--help, --h`	Muestra la ayuda sobre este comando en la línea de comandos.
`--silent, --s`	Muestra solamente los mensajes de error. El valor predeterminado es **false**.
`--verbose, --v`	Muestra información suplementaria durante la ejecución. El valor predeterminado es **false**.

Ejemplo

Este comando actualiza la lista de esquemas:

```
xmlschemamanager update
```

8.5.9 upgrade

Este comando actualiza todos los esquemas aptos para la versión *parche* más reciente disponible. Puede identificar cuáles lo son con el comando `list -u`.

Nota: el comando `upgrade` eliminaría un esquema obsoleto si no hay ninguna versión disponible.

Sintaxis

```
<exec> upgrade [opciones]
```

Opciones
Estas son las opciones del comando `upgrade`:

`--help, --h`	Muestra la ayuda sobre este comando en la línea de comandos.
`--silent, --s`	Muestra solamente los mensajes de error. El valor predeterminado es **`false`**.
`--verbose, --v`	Muestra información suplementaria durante la ejecución. El valor predeterminado es **`false`**.

© 2017-2023 Altova GmbH

9 Información adicional

Esta sección contiene información adicional sobre:

- Códigos de salida 402
- Sugerencias sobre ubicación de esquemas 403

9.1 Códigos de salida

Estos son los códigos de salida disponibles:

0	La validación finalizó correctamente
1	La validación terminó con errores / Proceso interrumpido por Ctrl+C/Pausa/terminal cerrada / La licencia expiró durante la ejecución
11	RaptorXML no se puede iniciar. El motivo aparece en el archivo de registro
22	No se puede cargar el catálogo raíz / No se puede cargar el archivo de lista
64	Comandos / opciones no válidos
77	Error al adquirir licencia al inicio
128+**n**	RaptorXML finalizó debido al número de señal **n**. Todos los códigos de salida superiores a **128** indican que la finalización se debe a la recepción de una señal externa o a una señal que se desencadenó internamente. Por ejemplo, si el código de salida es **134**, el número de señal será **134-128=6** (el número de **SIGABRT**).

© 2017-2023 Altova GmbH

9.2 Sugerencias sobre ubicación de esquemas

Los documentos de instancia pueden usar sugerencias para indicar la ubicación del esquema. Para estas sugerencias se utilizan dos atributos:

- `xsi:schemaLocation` para documentos de esquema con espacios de nombres de destino. El valor de atributos es un par de elementos: el primero de ellos es un espacio de nombres y el segundo es una URL que encuentra el documento de esquema. El nombre del espacio de nombres debe coincidir con el espacio de nombres de destino del documento de esquema.
```
<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` para documentos de esquema sin espacios de nombres de destino. El valor de atributos es la URL del documento de esquema. El documento de esquema al que se hace referencia no puede tener un espacio de nombres de destino.
```
<document xmlns="http://www.altova.com/schemas/test03"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:noNamespaceSchemaLocation="Test.xsd">
```

La opción `--schemalocation-hints` especifica cómo usar estos atributos en las sugerencias y, sobre todo, cómo utilizar la información del atributo `schemaLocation` (vers descripción de esta opción más arriba). Recuerde que RaptorXML Server considera que la parte del espacio de nombres del valor `xsi:noNamespaceSchemaLocation` es una cadena vacía.

Las sugerencias sobre la ubicación de los esquemas también se puede dar en una instrucción de importación `import` de un documento XML Schema.

```
<import namespace="someNS" schemaLocation="someURL">
```

En la instrucción de importación las sugerencias también se pueden dar mediante un espacio de nombres que se pueda asignar a un esquema del archivo de catálogo o mediante una URL directamente (en el atributo `schemaLocation`). La opción `--schema-imports` 240 (para XBRL y XSD/XML) indica cómo se debe seleccionar la ubicación del esquema.

10 Información sobre motores

Este apartado contiene información sobre los motores XSLT y XQuery que contiene RaptorXML Server. Esa información se refiere principalmente al comportamiento de los motores en situaciones en las que este no viene determinado por las especificaciones, sino que depende de la implementación. Este apartado también contiene información sobre las funciones extendidas de Altova para XPath/XQuery.

© 2017-2023 Altova GmbH

10.1 Información sobre motores XSLT y XQuery

Los motores XSLT y XQuery de RaptorXML Server siguen las especificaciones del W3C y, por tanto, son más estrictos que otros motores anteriores de Altova, como los de las versiones antiguas de XMLSpy y del predecesor de RaptorXML Server, el procesador descatalogado AltovaXML. Por consiguiente, RaptorXML Server señala algunos errores leves que antes no se notificaban en la versión anterior de estos motores.

Por ejemplo:

- Se notifica un error de tipo (`err:XPTY0018`) si el resultado de un operador de ruta de acceso contiene tanto nodos como no nodos.
- Se notifica un error de tipo (`err:XPTY0019`) si `E1` en una expresión XPath `E1/E2` no da como resultado una secuencia de nodos.

Si encuentra este tipo de errores, modifique el documento XSLT/XQuery o el documento de instancia según corresponda.

Esta sección describe características relacionadas con la implementación de los motores e incluye estos apartados:

- XSLT 1.0 405
- XSLT 2.0 405
- XSLT 3.0 407
- XQuery 1.0 408
- XQuery 3.1 412

10.1.1 XSLT 1.0

El motor XSLT 1.0 de RaptorXML Server cumple con la recomendación XSLT 1.0 del 16 de noviembre de 1999 y con la recomendación XPath 1.0 del 16 de noviembre de 1999, ambas del W3C.

Nota sobre la implementación

Cuando el atributo `method` de `xsl:output` tiene el valor HTML o si selecciona de forma predeterminada el formato de salida HTML, los caracteres especiales del archivo XML o XSLT se insertan en el documento HTML como referencias de caracteres HTML. Por ejemplo, el carácter U+00A0 (la referencia de carácter hexadecimal para un espacio de no separación) se inserta en el código HTML como referencia de carácter (` ` o ` `) o como referencia de entidad (` `).

10.1.2 XSLT 2.0

Temas de este apartado:

- Especificaciones con las que cumple el motor 406
- Compatibilidad con versiones antiguas 406
- Espacios de nombres 406

- Compatibilidad con esquemas 407
- Comportamiento propio de esta implementación 407

Especificaciones

El motor XSLT 2.0 de RaptorXML Server cumple con la recomendación XSLT 2.0 del 23 de enero de 2007 y la recomendación XPath 2.0 del 14 de diciembre de 2010, ambas del W3C.

Compatibilidad con versiones antiguas

El motor XSLT 2.0 es compatible con versiones previas. Esto es relevante cuando se utiliza el motor XSLT 2.0 (parámetro de la interfaz de la línea de comandos `--xslt=2` 245) para procesar una hoja de estilos o instrucción XSLT 1.0. Tenga en cuenta que los resultados obtenidos con el motor XSLT 1.0 pueden ser diferentes a los obtenidos con el motor XSLT 2.0 en modo de compatibilidad con versiones antiguas.

Espacios de nombres

En su hoja de estilos XSLT 2.0 debe declarar estos espacios de nombres para poder usar los constructores de tipo y las funciones disponibles en XSLT 2.0. Los prefijos que aparecen a continuación son los que se suelen usar, pero puede usar otros prefijos si quiere.

Espacio de nombres	Prefijo	URI del espacio de nombres
Tipos XML Schema	`xs:`	`http://www.w3.org/2001/XMLSchema`
Funciones XPath 2.0	`fn:`	`http://www.w3.org/2005/xpath-functions`

Estos espacios de nombres se suelen declarar en el elemento `xsl:stylesheet` o en el elemento `xsl:transform`:

```
<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>
```

Es necesario tener en cuenta que:

- El motor XSLT 2.0 utiliza el espacio de nombres Funciones XPath 2.0 y XQuery 1.0 como **espacio de nombres de funciones predeterminado**. Esto significa que puede usar funciones XPath 2.0 y XSLT 2.0 en su hoja de estilos sin prefijos. Si declara el espacio de nombres Funciones XPath 2.0 en su hoja de estilos con un prefijo, podrá usar el prefijo asignado en la declaración.
- Cuando se usan constructores de tipo y tipos del espacio de nombres XML Schema, el prefijo utilizado en la declaración de espacio de nombres se debe usar en la llamada al constructor de tipo (por ejemplo, `xs:date`).
- Algunas funciones XPath 2.0 se llaman igual que algunos tipos de datos de XML Schema. Por ejemplo, las funciones XPath `fn:string` y `fn:boolean` y los tipos de datos de XML Schema `xs:string` y `xs:boolean`. Por tanto, si usa la expresión `string('Hello')`, la expresión se evalúa como `fn:string('Hello')` y no como `xs:string('Hello')`.

© 2017-2023 Altova GmbH

Compatibilidad con esquemas

El motor XSLT 2.0 está preparado para esquemas de modo que puede usar tipos de esquema definidos por el usuario y la instrucción `xsl:validate`.

Comportamiento propio de esta implementación

Más abajo puede ver cómo se ocupa el motor XSLT 2.0 de algunos aspectos de algunas de las funciones XSLT 2.0 relacionadas con esta implementación.

`xsl:result-document`
También son compatibles estas codificaciones específicas de Altova: `x-base16tobinary` y `x-base64tobinary`.

`function-available`
Esta función mira si hay funciones del ámbito disponibles (funciones XSLT, XPath y de extensión).

`unparsed-text`
El atributo `href` acepta (i) rutas de acceso relativas para archivos que estén en la carpeta del URI base y (ii) rutas de acceso absolutas con o sin el protocolo `file://`. También son compatibles estas codificaciones específicas de Altova: `x-binarytobase16` y `x-binarytobase64`. Ejemplo: `xs:base64Binary(unparsed-text('chart.png', 'x-binarytobase64'))`.

`unparsed-text-available`
El argumento `href` acepta (i) rutas de acceso relativas para archivos que estén en la carpeta del URI base y (ii) rutas de acceso absolutas con o sin el protocolo `file://`. También son compatibles estas codificaciones específicas de Altova: `x-binarytobase16` y `x-binarytobase64`.

Nota: estos valores de codificación estaban implementados en el ya descatalogado AltovaXML pero ya no se utilizan (son obsoletos): `base16tobinary`, `base64tobinary`, `binarytobase16` y `binarytobase64`.

10.1.3 XSLT 3.0

El motor XSLT 3.0 de RaptorXML Server cumple con la propuesta de recomendación XSLT 3.0 del 8 de junio de 2017 y con la propuesta de recomendación XPath 3.1 del 21 de marzo de 2017 del consorcio W3C.

El motor XSLT 3.0 tiene las mismas características de implementación que el motor XSLT 2.0 405. Pero además ofrece compatibilidad con muchas de las nuevas funciones XSLT3.0, con las funciones y los operadores XPath/XQuery 3.1 y con la especificación XPath 3.1.

Nota: la característica opcional de transmisión por secuencias no es compatible por ahora. Todo el documento se cargará en memoria independientemente del valor del atributo `streamable` y se procesará si hay suficiente memoria. Si no hay suficiente memoria, (i) se procesa todo el documento sin transmisión de secuencias, (ii) se procesan los constructores "guaranteed-streamable" como si se estuviera usando transmisión por secuencias y (iii) los errores de transmisión de secuencias no se reconocen. En las aplicaciones de 64 bits la ejecución sin transmisión no debería causar problemas. Sin embargo, si se dan problemas de memoria, una solución sería añadir más memoria al sistema.

Espacios de nombres

En su hoja de estilos XSLT 3.0 debe declarar estos espacios de nombres para poder usar todos los constructores de tipo y las funciones disponibles en XSLT 3.0. Los prefijos que aparecen a continuación son los que se suelen usar, pero puede usar otros prefijos si quiere.

Espacio de nombres	Prefijo	URI del espacio de nombres
Tipos XML Schema	`xs:`	`http://www.w3.org/2001/XMLSchema`
Funciones XPath/XQuery 3.1	`fn:`	`http://www.w3.org/2005/xpath-functions`
Funciones matemáticas	`math:`	`http://www.w3.org/2005/xpath-functions/math`
Funciones de asignación	`map:`	`http://www.w3.org/2005/xpath-functions/map`
Funciones de matriz	`array:`	`http://www.w3.org/2005/xpath-functions/array`
Códigos de error Xquery, XSLT y XPath	`err:`	`http://www.w3.org/2005/xpath-functions/xqt-errors`
Funciones de serialización	`output`	`http://www.w3.org/2010/xslt-xquery-serialization`

Por lo general, estos espacios de nombres se declaran en el elemento `xsl:stylesheet` o `xsl:transform`, como se puede ver en este extracto:

```
<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>
```

Es necesario tener en cuenta que:

- El motor XSLT 3.0 utiliza el espacio de nombres Funciones y operadores XPath/XQuery 3.1 como **espacio de nombres de funciones predeterminado**. Esto significa que puede usar funciones de este espacio de nombres en su hoja de estilos sin ningún prefijo. Si declara el espacio de nombres Funciones en su hoja de estilos con un prefijo, podrá usar el prefijo asignado en la declaración.
- Cuando se usan constructores de tipo y tipos del espacio de nombres XML Schema, el prefijo utilizado en la declaración de espacio de nombres se debe usar en la llamada al constructor de tipo (por ejemplo, `xs:date`).
- Algunas funciones XPath/XQuery se llaman igual que algunos tipos de datos de XML Schema. Por ejemplo, las funciones XPath `fn:string` y `fn:boolean` y los tipos de datos de XML Schema `xs:string` y `xs:boolean`. Por tanto, si usa la expresión `string('Hello')`, la expresión se evalúa como `fn:string('Hello')` y no como `xs:string('Hello')`.

10.1.4 XQuery 1.0

Temas de este apartado:

© 2017-2023 Altova GmbH

- Especificaciones con las que cumple el motor 409
- Compatibilidad con esquemas 409
- Codificación 409
- Espacios de nombres 409
- Fuentes XML y validación 410
- Comprobación de tipos estática y dinámica 410
- Módulos biblioteca 410
- Funciones externas 411
- Intercalaciones 411
- Precisión de datos numéricos 411
- Compatibilidad con instrucciones XQuery 411
- Comportamiento propio de esta implementación 411

Especificaciones compatibles

El motor XQuery 1.0 de RaptorXML Server cumple con la recomendación XQuery 1.0 del 14 de diciembre de 2010 del W3C. El estándar XQuery concede libertad a la hora de implementar muchas características. A continuación explicamos cómo se implementaron estas características en el motor XQuery 1.0 de RaptorXML Server.

Compatibilidad con esquemas

El motor XQuery 1.0 está preparado para esquemas.

Codificación

El motor XQuery 1.0 es compatible con las codificaciones de caracteres UTF-8 y UTF-16.

Espacios de nombres

Se predefinen estos URI de espacios de nombres y sus enlaces asociados.

Espacio de nombres	Prefijo	URI del espacio de nombres
Tipos XML Schema	`xs:`	`http://www.w3.org/2001/XMLSchema`
Schema instance	`xsi:`	`http://www.w3.org/2001/XMLSchema-instance`
Funciones integradas	`fn:`	`http://www.w3.org/2005/xpath-functions`
Funciones locales	`local:`	`http://www.w3.org/2005/xquery-local-functions`

Es importante tener en cuenta que:

- El motor XQuery 1.0 entiende que los prefijos de la tabla anterior están enlazados con los correspondientes espacios de nombres.
- Como el espacio de nombres de funciones integradas (véase `fn:`) es el espacio de nombres de funciones predeterminado de XQuery, no es necesario usar el prefijo `fn:` cuando se invocan funciones integradas (por ejemplo, `string("Hello")` llamará a la función `fn:string`). No obstante, el prefijo `fn:` se puede utilizar para llamar a una función integrada sin necesidad de declarar el espacio de nombres en el prólogo de la consulta (por ejemplo: `fn:string("Hello")`).

- Puede cambiar el espacio de nombres de funciones predeterminado declarando la expresión `default function namespace` en el prólogo de la consulta.
- Cuando use tipos del espacio de nombres XML Schema, puede usar el prefijo `xs:` sin necesidad de declarar los espacios de nombres de forma explícita ni enlazar estos prefijos a los espacios de nombres en el prólogo de la consulta. (Ejemplo: `xs:date` y `xs:yearMonthDuration`.) Si quiere usar otros prefijos para el espacio de nombres de XML Schema, estos se deben declarar en el prólogo de la consulta. (Ejemplo: `declare namespace alt = "http://www.w3.org/2001/XMLSchema"; alt:date("2004-10-04").`)
- Recuerde que los tipos de datos `untypedAtomic`, `dayTimeDuration` y `yearMonthDuration` se movieron del espacio de nombres XPath Datatypes al espacio de nombres XML Schema (es decir, ahora es `xs:yearMonthDuration`.)

Si se asignaron mal los espacios de nombres para funciones, constructores de tipo, pruebas de nodo, etc., se emite un error. Sin embargo, recuerde que algunas funciones se llaman igual que los tipos de datos de esquema (p. ej. `fn:string` y `fn:boolean`.) (Se definen `xs:string` y `xs:boolean`.) El prefijo del espacio de nombres determina si se usa la función o el constructor de tipo.

Documento XML de origen y validación

Los documentos XML que se utilizan para ejecutar un documento XQuery con el motor XQuery 1.0 deben tener un formato XML correcto. Sin embargo, no es necesario que sean válidos con respecto a un esquema XML. Si el archivo no es válido, el archivo no válido se carga sin información de esquema. Si el archivo XML está asociado a un esquema externo y es válido con respecto a dicho esquema, se genera información posterior a la validación de esquema, que se utilizará para evaluar la consulta.

Comprobación de tipos estática y dinámica

En la fase de análisis estático se revisan aspectos de la consulta como la sintaxis, si existen referencias externas (p. ej. para módulos), si las funciones y variables que se invocan están definidas, etc. Si se detecta un error en la fase de análisis estático, se notifica y la ejecución se interrumpe.

La comprobación dinámica de tipos se realiza en tiempo de ejecución, cuando la consulta se ejecuta. Si un tipo no es compatible con los requisitos de una operación, se emite un error. Por ejemplo, la expresión `xs:string("1") + 1` devuelve un error porque la operación de suma no se puede llevar a cabo en un operando de tipo `xs:string`.

Módulos biblioteca

Los módulos biblioteca almacenan funciones y variables para poder volver a utilizarlas. El motor XQuery 1.0 es compatible con el uso de módulos almacenados en un **solo archivo XQuery externo**. Dicho archivo de módulo debe incluir una declaración `module` en su prólogo que apunte a un espacio de nombres de destino Por ejemplo:

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

Todas las funciones y variables declaradas en el módulo pertenecen al espacio de nombres asociado al módulo. El módulo se importa en un archivo XQuery con la instrucción `import module` del prólogo de la consulta. La instrucción `import module` solamente importa funciones y variables declaradas directamente en el archivo de módulo biblioteca. Por ejemplo:

```
import module namespace modlib = "urn:module-library" at "modulefilename.xq";
```

 © 2017-2023 Altova GmbH

```
if        ($modlib:company = "Altova")
then       modlib:webaddress()
else       error("No match found.")
```

Funciones externas

Las funciones externas son incompatibles con el motor XQuery 1.0, es decir, todas las expresiones que usen la palabra clave `external`. Por ejemplo:

```
declare function hoo($param as xs:integer) as xs:string external;
```

Intercalaciones

La intercalación predeterminada es la intercalación de puntos de código Unicode, que compara las cadenas de texto según sus puntos de código Unicode. Otras intercalaciones compatibles son las intercalaciones ICU que se enumeran aquí[414]. Para usar una intercalación concreta, indique su URI tal y como aparece en la lista de intercalaciones compatibles[414]. Las comparaciones de cadenas de texto, incluidas las comparaciones para las funciones `fn:max` y `fn:min`, se harán según la intercalación especificada. Si no se indica la opción de intercalación, se utiliza la intercalación de puntos de código Unicode predeterminada.

Precisión de tipos numéricos

- El tipo de datos `xs:integer` es de precisión arbitraria, es decir, puede representar un número de dígitos cualquiera.
- El tipo de datos `xs:decimal` tiene un límite de 20 dígitos después del punto decimal.
- Los tipos de datos `xs:float` y `xs:double` tienen una precisión limitada de 15 dígitos.

Compatibilidad con instrucciones XQuery

La instrucción `Pragma` no es compatible. Si se encuentra, se ignora y en su lugar se evalúa la expresión de reserva.

Comportamiento propio de esta implementación

A continuación puede ver una descripción de cómo enfocan los motores XQuery y XQuery Update 1.0 los aspectos relativos a la implementación de ciertas funciones.

`unparsed-text`

El atributo `href` acepta (i) rutas de acceso relativas para archivos que estén en la carpeta del URI base y (ii) rutas de acceso absolutas con o sin el protocolo `file://`. También son compatibles estas codificaciones específicas de Altova: `x-binarytobase16` y `x-binarytobase64`. Ejemplo: `xs:base64Binary(unparsed-text('chart.png', 'x-binarytobase64'))`.

`unparsed-text-available`

El argumento `href` acepta (i) rutas de acceso relativas para archivos que estén en la carpeta del URI base y (ii) rutas de acceso absolutas con o sin el protocolo `file://`. También son compatibles estas codificaciones específicas de Altova: `x-binarytobase16` y `x-binarytobase64`.

Nota: estos valores de codificación estaban implementados en el ya descatalogado AltovaXML pero ya no se utilizan (son obsoletos): `base16tobinary`, `base64tobinary`, `binarytobase16` y `binarytobase64`.

10.1.5 XQuery 3.1

El motor XQuery 3.1 de RaptorXML Server cumple con la propuesta de recomendación XQuery 3.1 del 21 de marzo de 2017 del consorcio W3C y es compatible con funciones XPath y XQuery 3.1. La especificación XQuery 3.1 es un supraconjunto de la especificación 3.0. El motor XQuery 3.1, por tanto, es compatible con las características de XQuery 3.0.

Espacios de nombres

En su documento XQuery 3.1 debe declarar estos espacios de nombres para poder usar todos los constructores de tipo y las funciones disponibles en XQuery 3.1. Los prefijos que aparecen a continuación son los que se suelen usar, pero puede usar otros prefijos si quiere.

Espacio de nombres	Prefijo	URI del espacio de nombres
Tipos XML Schema	`xs:`	`http://www.w3.org/2001/XMLSchema`
Funciones XPath/XQuery 3.1	`fn:`	`http://www.w3.org/2005/xpath-functions`
Funciones matemáticas	`math:`	`http://www.w3.org/2005/xpath-functions/math`
Funciones de asignación	`map:`	`http://www.w3.org/2005/xpath-functions/map`
Funciones de matriz	`array:`	`http://www.w3.org/2005/xpath-functions/array`
Códigos de error Xquery, XSLT y XPath	`err:`	`http://www.w3.org/2005/xpath-functions/xqt-errors`
Funciones de serialización	`output`	`http://www.w3.org/2010/xslt-xquery-serialization`

Es necesario tener en cuenta que:

- El motor XQuery 3.1 entiende que los prefijos de la tabla anterior están enlazados con los correspondientes espacios de nombres.
- Como el espacio de nombres de funciones integradas (véase `fn:`) es el espacio de nombres de funciones predeterminado de XQuery, no es necesario usar el prefijo `fn:` cuando se invocan funciones integradas (por ejemplo, `string("Hello")` llamará a la función fn:string). No obstante, el prefijo `fn:` se puede utilizar para llamar a una función integrada sin necesidad de declarar el espacio de nombres en el prólogo de la consulta (por ejemplo: `fn:string("Hello")`).
- Puede cambiar el espacio de nombres de funciones predeterminado declarando la expresión `default function namespace` en el prólogo de la consulta.
- Cuando use tipos del espacio de nombres XML Schema, puede usar el prefijo `xs:` sin necesidad de declarar los espacios de nombres de forma explícita ni enlazar estos prefijos a los espacios de nombres en el prólogo de la consulta. (Ejemplo: `xs:date` y `xs:yearMonthDuration`.) Si quiere usar otros prefijos para el espacio de nombres de XML Schema, estos se deben declarar en el prólogo de la consulta. (Ejemplo: `declare namespace alt = "http://www.w3.org/2001/XMLSchema"; alt:date("2004-10-04")`.)

Si se asignaron mal los espacios de nombres para funciones, constructores de tipo, pruebas de nodo, etc., se emite un error. Sin embargo, recuerde que algunas funciones se llaman igual que los tipos de datos de

 © 2017-2023 Altova GmbH

esquema (p. ej. `fn:string` y `fn:boolean`.) (Se definen `xs:string` y `xs:boolean`.) El prefijo del espacio de nombres determina si se usa el constructor de funciones o el de tipos.

Comportamiento propio de esta implementación

Tiene las mismas características de implementación que el motor XQuery 1.0 [408].

Además, el cifrado de Altova `x-base64tobinary` se puede usar para crear un documento de resultados binario, como una imagen.

10.2 Funciones XSTL y XPath/XQuery

Esta sección enumera las funciones de extensión de Altova y otras funciones de extensión que se pueden utilizar con expresiones XPath y XQuery. Las funciones de extensión de Altova se pueden usar con los motores XSLT y XQuery de Altova y ofrecen algunas funciones más aparte de las que están disponibles en las bibliotecas de funciones definidas en los estándares del W3C.

En esta sección describimos principalmente las funciones de extensión XPath/XQuery que han sido creadas por Altova para proporcionar operaciones adicionales. Estas funciones 415 pueden ser calculadas por los motores XSLT y XQuery de Altova basándose en las reglas descritas en esta sección. Para obtener información sobre las funciones XPath/XQuery regulares, consulte la Referencia de funciones XPath/XQuery de Altova.

Aspectos generales

Es necesario tener en cuenta estos puntos generales:

- A las funciones de las bibliotecas de funciones principales definidas en las especificaciones W3C se les puede llamar sin un prefijo. Esto se debe a que los motores XSLT y XQuery leen funciones sin prefijo como si pertenecieran a un espacio de nombres de funciones predeterminado `http://www.w3.org/2005/xpath-functions`, que es el que se especifica en las especificaciones de las funciones XPath y XQuery. Si este espacio de nombres se declara explícitamente en un documento XSLT o XQuery, el prefijo utilizado en la declaración de espacio de nombres también se puede usar en el nombre de las funciones.
- Por lo general, si una función espera como argumento una secuencia de un elemento y se suministra una secuencia de más de un elemento, entonces se devuelve un error.
- Se usa la colación de punto de código de Unicode para todas las comparaciones de cadenas de texto.
- Los resultados que son QName se serializan de esta forma `[prefijo:]nombrelocal`.

Precisión de xs:decimal

La precisión se refiere a la cantidad de dígitos del número; la especificación requiere un mínimo de 18 dígitos. Para operaciones de división que dan un resultado de tipo `xs:decimal`, la precisión es de 19 dígitos tras el punto decimal sin redondeos.

Zona horaria implícita

Cuando hay que comparar dos valores `date`, `time` o `dateTime`, es necesario conocer el uso horario de los valores que se deben comparar. Cuando el uso horario no se conoce de forma explícita, se usa el uso horario implícito. La zona horaria implícita se toma del reloj del sistema y para probar cuál es su valor puede utilizar la función `implicit-timezone()`.

Intercalaciones

La colación predeterminada es la colación de punto de código de Unicode, que compara cadenas de texto basándose en su punto de código. El motor usa el algoritmo de colación de Unicode. Otras intercalaciones compatibles son las intercalaciones ICU que aparecen más abajo. Para usar una intercalación indique su URI tal y como aparece en la tabla más abajo. Las comparaciones de cadenas de texto (incluidas las que usan las funciones `max` y `min`) se harán según la intercalación especificada. Si no se ha indicado ninguna colación se usará la colación predeterminada de punto de código de Unicode.

 © 2017-2023 Altova GmbH

Lenguaje	URIs
da: Danés	da_DK
de: Alemán	de_AT, de_BE, de_CH, de_DE, de_LI, de_LU
en: Inglés	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: Español	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: Francés	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: Italiano	it_CH, it_IT
ja: Japonés	ja_JP
nb: Noruego bokmål	nb_NO
nl: Holandés	nl_AW, nl_BE, nl_NL
nn: Noruego nynorsk	nn_NO
pt: Portugués	pt_AO, pt_BR, pt_GW, pt_MZ, pt_PT, pt_ST
ru: Ruso	ru_MD, ru_RU, ru_UA
sv: Sueco	sv_FI, sv_SE

Eje del espacio de nombres
El eje del espacio de nombres está obsoleto en XPath 2.0. Sin embargo, sí que se admite el uso del espacio de nombres. Para acceder a la información sobre el espacio de nombres con mecanismos de XPath 2.0, utilice las funciones `in-scope-prefixes()`, `namespace-uri()` y `namespace-uri-for-prefix()`.

10.2.1 Funciones de extensión de Altova

Puede utilizar todas las funciones de extensión de Altova en sus expresiones XPath/XQuery. Con ellas conseguirá funciones adicionales no disponibles en la biblioteca de funciones estándar de XPath, XQuery y XSLT. Las funciones de extensión de Altova están en el **espacio de nombres** **`http://www.altova.com/xslt-extensions`** y en esta sección se presentan con el prefijo, que se supone estará enlazado al espacio de nombres señalado. Tenga en cuenta que en futuras versiones del producto algunas funciones pueden dejar de ser compatibles o su comportamiento puede cambiar. Por tanto, consulte siempre la documentación del producto para conocer el funcionamiento de estas funciones en cada versión del producto.

Las funciones definidas en las especificaciones XPath/XQuery Functions del W3C se pueden usar en (i) expresiones XPath en contextos XSLT y en (ii) expresiones XQuery en documentos XQuery. En esta documentación las funciones que se pueden usar en el primer contexto (XPath en XSLT) llevan el símbolo **XP** y se les llama funciones XPath. Las funciones que se pueden usar en contextos XQuery llevan el símbolo **XQ** y funcionan como funciones XQuery. Las especificaciones XSLT del W3C también definen funciones que se pueden usar en expresiones XPath en documentos XSLT. Estas funciones llevan el símbolo **XSLT** y se les denomina funciones XSLT. Por cada función se indica en qué versión de XPath/XQuery y XSLT se puede usar (ver símbolos más abajo). Las funciones de las bibliotecas de funciones XPath/XQuery y XSLT aparecen sin prefijo. Las funciones de extensión de otras bibliotecas, como las funciones de extensión de Altova, aparecen con un prefijo.

Funciones XPath (en expresiones XPath en XSLT):	**XP1** **XP2** **XP3.1.1**
Funciones XSLT (en expresiones XPath en XSLT):	**XSLT1** **XSLT2** **XSLT3**
Funciones XQuery (en expresiones XQuery en XQuery):	**XQ1** **XQ3.1**

Cómo usar las funciones de extensión de Altova

Para poder usar las funciones de extensión de Altova debe declarar el espacio de nombre correspondiente (*el primer resaltado en el extracto de código siguiente*) y después usar las funciones de extensión para que se resuelvan como si pertenecieran a ese espacio de nombres (*véase el segundo resaltado*). En el ejemplo siguiente puede ver cómo se usa la función de extensión de Altova **age**.

```
<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>
```

Funciones XSLT 417

Las funciones XSLT solo se pueden utilizar en expresiones XPath en un contexto XSLT (igual que las funciones XSLT 2.0 `current-group()` o `key()`). Estas funciones no están pensadas para contextos no XSLT (p. ej. contextos XQuery) y, por tanto, no funcionarán en contextos que no sean XSLT. Recuerde que las funciones XSLT para XBRL solamente se pueden utilizar con ediciones de los productos de Altova compatibles con XBRL.

© 2017-2023 Altova GmbH

Funciones XPath/XQuery

Las funciones XPath/XQuery se pueden utilizar en expresiones XPath, en contextos XSLT y en expresiones XQuery:

- Funciones de fecha y hora 420
- Funciones de geoubicación 438
- Funciones relacionadas con imágenes 449
- Funciones numéricas 454
- Funciones de secuencia 477
- Funciones de cadena 485
- Funciones varias 493

Funciones para gráficos (solo en las ediciones Enterprise y Server Edition)

Las funciones de extensión para gráficos de Altova 495 son compatibles con las ediciones Enterprise y Server Edition de los productos de Altova solamente. Estas funciones permiten generar gráficos a partir de datos XML.

Funciones para códigos de barras

Las funciones de extensión para códigos de barras de Altova 508 permiten generar códigos de barras y colocarlos en los resultados generados con hojas de estilos XSLT.

10.2.1.1 Funciones XSLT

Las **funciones de extensión XSLT** pueden utilizarse en expresiones XPath en contextos XSLT y no funcionan en contextos que no sean XSLT (por ejemplo, en contextos XQuery).

Nota sobre el nombre de las funciones y lenguajes

Puede utilizar todas las funciones de extensión de Altova en sus expresiones XPath/XQuery. Con ellas conseguirá funciones adicionales no disponibles en la biblioteca de funciones estándar de XPath, XQuery y XSLT. Las funciones de extensión de Altova están en el **espacio de nombres** `http://www.altova.com/xslt-extensions` y en esta sección se presentan con el prefijo, que se supone estará enlazado al espacio de nombres señalado. Tenga en cuenta que en futuras versiones del producto algunas funciones pueden dejar de ser compatibles o su comportamiento puede cambiar. Por tanto, consulte siempre la documentación del producto para conocer el funcionamiento de estas funciones en cada versión del producto.

Funciones XPath (en expresiones XPath en XSLT):	XP1 XP2 XP3.1.1
Funciones XSLT (en expresiones XPath en XSLT):	XSLT1 XSLT2 XSLT3
Funciones XQuery (en expresiones XQuery en XQuery):	XQ1 XQ3.1

Funciones generales

▼ distinct-nodes [altova:]

altova:distinct-nodes(*node()**) como **node()*** XSLT1 XSLT2 XSLT3
Toma un conjunto de nodos como entrada y devuelve el mismo conjunto menos los nodos que tengan el mismo valor (es decir, devuelve los nodos que son únicos). La comparación se hace con la función XPath/XQuery `fn:deep-equal`.

▣ *Ejemplo*

- **altova:distinct-nodes**(country) devuelve todos los nodos secundarios `country` excepto los que tengan el mismo valor.

▼ evaluate [altova:]

altova:evaluate(ExpresiónXPath *como xs:string***[, ValorDe$p1, ... ValorDe$pN])** XSLT1 XSLT2 XSLT3
Toma una expresión XPath, pasada como cadena, como argumento obligatorio. Devuelve el resultado de la expresión evaluada. Por ejemplo, **evaluate('//Name[1]')** devuelve el contenido del primer elemento `Name` del documento. Observe que para pasar la expresión `//Name[1]` como cadena basta con ponerla entre comillas simples.

La función `altova:evaluate` puede tomar más argumentos, que son los valores de las variables del ámbito que se llaman `p1`, `p2`, `p3`... `pN`. Recuerde que (i) las variables deben definirse con nombres de tipo `pX`, siendo `X` un entero; (ii) los argumentos de la función `altova:evaluate` (*ver firma más abajo*), a partir del segundo argumento, ofrecen los valores de las variables, correspondiendo la secuencia de argumentos a la secuencia numérica de variables: `p1` corresponde a `pN` y el segundo argumento será el valor de la variable `p1`, el tercer argumento al de la variable `p2`, y así sucesivamente; (iii) los valores de las variables deben ser de tipo `item*`.

▣ *Ejemplo*

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

En el ejemplo anterior puede observar que:

- El segundo argumento de la expresión `altova:evaluate` es el valor asignado a la variable `$p1`, el tercer argumento es el valor asignado a la variable `$p2` y así sucesivamente.
- Observe que el cuarto argumento de la función es un valor de cadena porque va entre comillas simples.
- El atributo `select` del elemento `xs:variable` suministra la expresión XPath. Como esta expresión debe ser de tipo `xs:string`, se pone entre comillas simples.

▣ *Más ejemplos*

-
```
<xsl:variable name="xpath" select="'$p1'" />
<xsl:value-of select="altova:evaluate($xpath, //Name[1])" />
```
El resultado es el valor del primer elemento `Name`.

 © 2017-2023 Altova GmbH

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

La función de extensión `altova:evaluate()` es muy práctica cuando una expresión XPath de la hoja de estilos XSLT contiene partes que se deben evaluar de forma dinámica. Por ejemplo, imagine que el usuario selecciona un criterio de ordenación y este criterio se almacena en el atributo `UserReq/@sortkey`. En la hoja de estilos podría tener esta expresión:

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

La función `altova:evaluate()` lee el atributo `sortkey` del elemento secundario `UserReq` del primario del nodo de contexto. Imagine que el valor del atributo `sortkey` es `Price`. En ese caso, la función `altova:evaluate()` devuelve `Price`, que se convierte en el valor del atributo `select`:

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

Si esta instrucción `sort` aparece dentro del contexto de un elemento llamado `Order`, entonces los elementos `Order` se ordenan según el valor de los secundarios `Price`. Otra opción es que, si el valor de `@sortkey` fuera `Date`, por ejemplo, entonces los elementos `Order` se ordenarían según el valor de los secundarios `Date`. Es decir, el criterio de ordenación para `Order` se selecciona del atributo `sortkey` en tiempo de ejecución. Esto no sería posible con una expresión como:

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

En este caso, el criterio de ordenación sería el propio atributo `sortkey`, no `Price` ni `Date` (ni otro contenido actual de `sortkey`).

Nota: el contexto estático incluye espacios de nombres, tipos y funciones (pero no variables) del entorno de llamada. El URI base y el espacio de nombres predeterminado se heredan.

Más ejemplos

- Variables estáticas: `<xsl:value-of select="$i3, $i2, $i1" />`
  El resultado es los valores de las tres variables.

- Expresión XPath dinámica con variables dinámicas:
  `<xsl:variable name="xpath" select="'$p3, $p2, $p1'" />`
  `<xsl:value-of select="altova:evaluate($xpath, 10, 20, 30)" />`
  El resultado es "`30 20 10`"

- Expresión XPath dinámica sin variables dinámicas:
  `<xsl:variable name="xpath" select="'$p3, $p2, $p1'" />`
  `<xsl:value-of select="altova:evaluate($xpath)" />`
  Error: no se definió la variable para $p3.

encode-for-rtf [altova:]

altova:encode-for-rtf(entrada *como xs:string*, **conservarEspaciosEnBlanco** *como xs:boolean*, **conservarLíneasNuevas** *como xs:boolean*) como **xs:string** XSLT2 XSLT3

Convierte la cadena de entrada en código para RTF. Los espacios en blanco y las líneas nuevas se conservan o no dependiendo del valor booleano especificado para los correspondientes parámetros.

[Subir 417]

Funciones XBRL

Las funciones XBRL de Altova solo funcionan en las ediciones de los productos de Altova que son compatibles con XBRL.

▼ xbrl-footnotes [altova:]

altova:xbrl-footnotes(*node()***)** como **node()*** XSLT2 XSLT3
Toma un nodo como argumento de entrada y devuelve el conjunto de nodos de nota al pie XBRL al que hace referencia el nodo de entrada.

▼ xbrl-labels [altova:]

altova:xbrl-labels(*xs:QName, xs:string***)** como **node()*** XSLT2 XSLT3
Toma dos argumentos de entrada: un nombre de nodo y la ubicación del archivo de taxonomía en el que está el nodo. La función devuelve los nodos de etiqueta XBRL asociados al nodo de entrada.

[Subir 417]

10.2.1.2 Funciones XPath/XQuery: de fecha y hora

Las funciones de extensión de fecha y hora de Altova se pueden usar en expresiones XPath y XQuery y permiten procesar datos almacenados en tipos de datos XML Schema de fecha y hora. Estas funciones se pueden usar con los **motores XPath 3.0 y XQuery 3.0** de Altova y están disponibles en contextos XPath/XQuery.

Nota sobre el nombre de las funciones y lenguajes

Puede utilizar todas las funciones de extensión de Altova en sus expresiones XPath/XQuery. Con ellas conseguirá funciones adicionales no disponibles en la biblioteca de funciones estándar de XPath, XQuery y XSLT. Las funciones de extensión de Altova están en el **espacio de nombres** **http://www.altova.com/xslt-extensions** y en esta sección se presentan con el prefijo, que se supone estará enlazado al espacio de nombres señalado. Tenga en cuenta que en futuras versiones del producto algunas funciones pueden dejar de ser compatibles o su comportamiento puede cambiar. Por tanto, consulte siempre la documentación del producto para conocer el funcionamiento de estas funciones en cada versión del producto.

Funciones XPath (en expresiones XPath en XSLT):	XP1 XP2 XP3.1.1
Funciones XSLT	XSLT1 XSLT2 XSLT3

 © 2017-2023 Altova GmbH

(en expresiones XPath en XSLT):	
Funciones XQuery *(en expresiones XQuery en XQuery):*	XQ1 XQ3.1

▼ Funciones agrupadas según su funcionalidad

- Agregar una duración a xs:dateTime y devolver xs:dateTime 422
- Agregar una duración a xs:date y devolver xs:date 424
- Agregar una duración a xs:time y devolver xs:time 425
- Recuperar duraciones y aplicarles formato 425
- Quitar la zona horaria de las funciones que generan la fecha/hora actual 426
- Devolver el número de días, horas, minutos y segundos de duraciones 428
- Devolver el día de la semana de una fecha como número entero 429
- Devolver el número de semana de una fecha como número entero 431
- Generar la fecha, la hora y el tipo de duración a partir de los componentes léxicos de cada tipo 432
- Construir un tipo date, dateTime o a partir de la cadena de entrada 433
- Funciones para calcular la edad 435
- Funciones para calcular el tiempo Unix 436

▼ Funciones por orden alfabético

altova:add-days-to-date 424
altova:add-days-to-dateTime 422
altova:add-hours-to-dateTime 422
altova:add-hours-to-time 425
altova:add-minutes-to-dateTime 422
altova:add-minutes-to-time 425
altova:add-months-to-date 424
altova:add-months-to-dateTime 422
altova:add-seconds-to-dateTime 422
altova:add-seconds-to-time 425
altova:add-years-to-date 424
altova:add-years-to-dateTime 422
altova:age 435
altova:age-details 435
altova:build-date 432
altova:build-duration 432
altova:build-time 432
altova:current-dateTime-no-TZ 426
altova:current-date-no-TZ 426
altova:current-time-no-TZ 426
altova:date-no-TZ 426
altova:dateTime-from-epoch 436
altova:dateTime-from-epoch-no-TZ 436
altova:dateTime-no-TZ 426
altova:days-in-month 428
altova:epoch-from-dateTime 436
altova:hours-from-dateTimeDuration-accumulated 428
altova:minutes-from-dateTimeDuration-accumulated 428
altova:seconds-from-dateTimeDuration-accumulated 428
altova:format-duration 425
altova:parse-date 433
altova:parse-dateTime 433

altova:parse-duration 425
altova:parse-time 433
altova:time-no-TZ 426
altova:weekday-from-date 429
altova:weekday-from-dateTime 429
altova:weeknumber-from-date 431
altova:weeknumber-from-dateTime 431

[Subir 420]

Agregar una duración a xs:dateTime XP3.1 XQ3.1

Estas funciones sirven para agregar una duración a **xs:dateTime** y devuelven **xs:dateTime**. El tipo xs:dateTime tiene el formato SSAA-MM-DDThh:mm:ss.sss. Se trata de la concatenación de los formatos xs:date y xs:time separados por la letra T. Si quiere puede usar un sufijo de zona horaria (por ejemplo **+01:00**).

add-years-to-dateTime [altova:]

altova:add-years-to-dateTime(FechaHora *as xs:dateTime*, **Años** *as xs:integer***)** como **xs:dateTime** XP3.1 XQ3.1

Añade una duración en años un valor de fecha y hora. El segundo argumento es el número de años que se debe añadir al valor de fecha y hora dado como primer argumento. El resultado es de tipo xs:dateTime.

Ejemplos

- **altova:add-years-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), 10) devuelve 2024-01-15T14:00:00
- **altova:add-years-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), -4) devuelve 2010-01-15T14:00:00

add-months-to-dateTime [altova:]

altova:add-months-to-dateTime(FechaHora *as xs:dateTime*, **Meses** *as xs:integer***)** como **xs:dateTime** XP3.1 XQ3.1

Añade una duración en meses a un valor de fecha y hora. El segundo argumento es el número de meses que se debe añadir al valor de fecha y hora dado como primer argumento. El resultado es de tipo xs:dateTime.

Ejemplos

- **altova:add-months-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), 10) devuelve 2014-11-15T14:00:00
- **altova:add-months-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), -2) devuelve 2013-11-15T14:00:00

add-days-to-dateTime [altova:]

altova:add-days-to-dateTime(FechaHora *as xs:dateTime*, **Días** *as xs:integer***)** como

 © 2017-2023 Altova GmbH

xs:dateTime XP3.1 XQ3.1
Añade una duración en días a un valor de fecha y hora. El segundo argumento es el número de días que se deben añadir al valor de fecha y hora dado como primer argumento. El resultado es de tipo `xs:dateTime`.

Ejemplos

- **altova:add-days-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), 10) devuelve 2014-01-25T14:00:00
- **altova:add-days-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), -8) devuelve 2014-01-07T14:00:00

add-hours-to-dateTime [altova:]

altova:add-hours-to-dateTime(FechaHora *as xs:dateTime*, **Horas** *as xs:integer***)** como **xs:dateTime** XP3.1 XQ3.1
Añade una duración en horas a un valor de fecha y hora. El segundo argumento es el número de horas que se deben añadir al valor de fecha y hora dado como primer argumento. El resultado es de tipo `xs:dateTime`.

Ejemplos

- **altova:add-hours-to-dateTime**(xs:dateTime("2014-01-15T13:00:00"), 10) devuelve 2014-01-15T23:00:00
- **altova:add-hours-to-dateTime**(xs:dateTime("2014-01-15T13:00:00"), -8) devuelve 2014-01-15T05:00:00

add-minutes-to-dateTime [altova:]

altova:add-minutes-to-dateTime(FechaHora *as xs:dateTime*, **Minutos** *as xs:integer***)** como **xs:dateTime** XP3.1 XQ3.1
Añade una duración en minutos a un valor de fecha y hora. El segundo argumento es el número de minutos que se debe añadir al valor de fecha y hora dado como primer argumento. El resultado es de tipo `xs:dateTime`.

Ejemplos

- **altova:add-minutes-to-dateTime**(xs:dateTime("2014-01-15T14:10:00"), 45) devuelve 2014-01-15T14:55:00
- **altova:add-minutes-to-dateTime**(xs:dateTime("2014-01-15T14:10:00"), -5) devuelve 2014-01-15T14:05:00

add-seconds-to-dateTime [altova:]

altova:add-seconds-to-dateTime(FechaHora *as xs:dateTime*, **Segundos** *as xs:integer***)** como **xs:dateTime** XP3.1 XQ3.1
Añade una duración en segundos a un valor de fecha y hora. El segundo argumento es el número de segundos que se debe añadir al valor de fecha y hora dado como primer argumento. El resultado es de tipo `xs:dateTime`.

Ejemplos

- `altova:add-seconds-to-dateTime(xs:dateTime("2014-01-15T14:00:10"), 20)` devuelve `2014-01-15T14:00:30`
- `altova:add-seconds-to-dateTime(xs:dateTime("2014-01-15T14:00:10"), -5)` devuelve `2014-01-15T14:00:05`

[Subir 420]

Agregar una duración a xs:date XP3.1 XQ3.1

Estas funciones agregan una duración a **xs:date** y devuelven **xs:date**. El tipo `xs:date` tiene el formato `SSAA-MM-DD`.

add-years-to-date [altova:]

altova:add-years-to-date(Fecha *as xs:date*, **Años** *as xs:integer***)** como **xs:date** XP3.1 XQ3.1
Añade una duración en años a una fecha. El segundo parámetro es el número de años que se debe añadir a la fecha dada como primer argumento. El resultado es de tipo `xs:date`.

Ejemplos

- `altova:add-years-to-date(xs:date("2014-01-15"), 10)` devuelve `2024-01-15`
- `altova:add-years-to-date(xs:date("2014-01-15"), -4)` devuelve `2010-01-15`

add-months-to-date [altova:]

altova:add-months-to-date(Fecha *as xs:date*, **Meses** *as xs:integer***)** como **xs:date** XP3.1 XQ3.1
Añade una duración en meses a una fecha. El segundo argumento es el número de meses que se debe añadir a la fecha dada como primer argumento. El resultado es de tipo `xs:date`.

Ejemplos

- `altova:add-months-to-date(xs:date("2014-01-15"), 10)` devuelve `2014-11-15`
- `altova:add-months-to-date(xs:date("2014-01-15"), -2)` devuelve `2013-11-15`

add-days-to-date [altova:]

altova:add-days-to-date(Fecha *as xs:date*, **Días** *as xs:integer***)** como **xs:date** XP3.1 XQ3.1
Añade una duración en días a una fecha. El segundo argumento es el número de días que se deben añadir a la fecha dad como primer argumento. El resultado es de tipo `xs:date`.

Ejemplos

- `altova:add-days-to-date(xs:date("2014-01-15"), 10)` devuelve `2014-01-25`
- `altova:add-days-to-date(xs:date("2014-01-15"), -8)` devuelve `2014-01-07`

[Subir 420]

 © 2017-2023 Altova GmbH

Recuperar duraciones y aplicarles formato XP3.1 XQ3.1

Estas funciones analizan la entrada **xs:duration** o **xs:string** y devuelven, respectivamente, **xs:string** o **xs:duration**.

▼ format-duration [altova:]

altova:format-duration(Duración *como xs:duration*, **Imagen** *como xs:string*) como **xs:string** XP3.1 XQ3.1
Aplica formato a una duración, que se suministra como primer argumento, en base a la cadena de imagen dada como segundo argumento. El resultado es una cadena de texto cuyo formato se ajusta a la cadena de imagen.

Ejemplos

- **altova:format-duration**(xs:duration("P2DT2H53M11.7S"), "Días:[D01] Horas:[H01] Minutos:[m01] Segundos:[s01] Fracciones:[f0]") devuelve "Días:02 Horas:02 Minutos:53 Segundos:11 Fracciones:7"
- **altova:format-duration**(xs:duration("P3M2DT2H53M11.7S"), "Meses:[M01] Días:[D01] Horas:[H01] Minutos:[m01]") devuelve "Meses:03 Días:02 Horas:02 Minutos:53"

▼ parse-duration [altova:]

altova:parse-duration(CadenaEntrada *como xs:string*, **Imagen** *como xs:string*) como **xs:duration** XP3.1 XQ3.1
Toma una cadena con patrón como primer argumento y una cadena de imagen como segundo argumento. La cadena de entrada se analiza en base a la cadena de imagen y se devuelve un xs:duration.

Ejemplos

- **altova:parse-duration**("Días:02 Horas:02 Minutos:53 Segundos:11 Fracciones:7"), "Días:[D01] Horas:[H01] Minutes:[m01] Segundos:[s01] Fracciones:[f0]") devuelve "P2DT2H53M11.7S"
- **altova:parse-duration**("Meses:03 Días:02 Horas:02 Minutos:53 Segundos:11 Fracciones:7", "Meses:[M01] Días:[D01] Horas:[H01] Minutos:[m01]") devuelve "P3M2DT2H53M"

[Subir 420]

Agregar una duración a xs:time XP3.1 XQ3.1

Estas funciones agregan una duración a **xs:time** y devuelven **xs:time**. El tipo xs:time tiene un formato léxico de este tipo hh:mm:ss.sss. Si quiere, puede añadir un sufijo de zona horaria. La letra Z indica (UTC). Las demás zonas horarias se representan con la diferencia que hay entre ellas y la zona UTC: +hh:mm o -hh:mm. Si falta el valor de zona horaria, se entiende que se desconoce (no se da por hecho que es UTC)

▼ add-hours-to-time [altova:]

altova:add-hours-to-time(Hora *as xs:time*, **Horas** *as xs:integer*) como **xs:time** XP3.1 XQ3.1
Añade una duración en horas a una hora. El segundo argumento es el número de horas que se debe añadir a la hora dada como primer argumento. El resultado es de tipo xs:time.

Ejemplos

- `altova:add-hours-to-time(xs:time("11:00:00"), 10)` devuelve `21:00:00`
- `altova:add-hours-to-time(xs:time("11:00:00"), -7)` devuelve `04:00:00`

add-minutes-to-time [altova:]

`altova:add-minutes-to-time(Hora` *as xs:time*, **`Minutos`** *as xs:integer*`)` como **`xs:time`** XP3.1 XQ3.1

Añade una duración en minutos a una hora. El segundo argumento es el número de minutos que se debe añadir a la hora dada como primer argumento. El resultado es de tipo `xs:time`.

Ejemplos

- `altova:add-minutes-to-time(xs:time("14:10:00"), 45)` devuelve `14:55:00`
- `altova:add-minutes-to-time(xs:time("14:10:00"), -5)` devuelve `14:05:00`

add-seconds-to-time [altova:]

`altova:add-seconds-to-time(Hora` *as xs:time*, **`Segundos`** *as xs:integer*`)` como **`xs:time`** XP3.1 XQ3.1

Añade una duración en segundos a una hora. El segundo argumento es el número de segundos que se debe añadir a la hora dada como primer argumento. El resultado es de tipo `xs:time`. El componente **`Segundos`** puede estar comprendido entre `0` y `59.999`.

Ejemplos

- `altova:add-seconds-to-time(xs:time("14:00:00"), 20)` devuelve `14:00:20`
- `altova:add-seconds-to-time(xs:time("14:00:00"), 20.895)` devuelve `14:00:20.895`

[Subir 420]

Quitar la parte de zona horaria de los tipos de datos date/time XP3.1 XQ3.1

Estas funciones quitan la zona horaria de los valores **`xs:dateTime`**, **`xs:date`** o **`xs:time`** actuales. Tenga en cuenta que la diferencia entre `xs:dateTime` y `xs:dateTimeStamp` es que en esta última la parte de zona horaria es obligatoria (mientras que en la primera es opcional). Es decir, el formato de un valor `xs:dateTimeStamp` puede ser `SSAA-MM-DDThh:mm:ss.sss±hh:mm` o `SSAA-MM-DDThh:mm:ss.sssZ`. Si la fecha y la hora se leen del reloj del sistema como `xs:dateTimeStamp`, la función `current-dateTime-no-TZ()` se puede usar para quitar la zona horaria.

current-date-no-TZ [altova:]

`altova:current-date-no-TZ()` como **`xs:date`** XP3.1 XQ3.1

Esta función no toma ningún argumento. Quita la parte de zona horaria de la función `current-date()` (que es la fecha actual según el reloj del sistema) y devuelve un valor de tipo `xs:date`.

Ejemplos

Si la fecha actual es **`2014-01-15+01:00`**:

 © 2017-2023 Altova GmbH

- **altova:current-date-no-TZ()** devuelve 2014-01-15

▼ current-dateTime-no-TZ [altova:]

altova:current-dateTime-no-TZ() como **xs:dateTime** XP3.1 XQ3.1
Esta función no toma ningún argumento. Quita la parte de zona horaria de current-dateTime() (que es la fecha y hora actual según el reloj del sistema) y devuelve un valor de tipo xs:dateTime.

⊟ *Ejemplos*

Si la fecha y hora actual es **2014-01-15T14:00:00+01:00**:

- **altova:current-dateTime-no-TZ()** devuelve 2014-01-15T14:00:00

▼ current-time-no-TZ [altova:]

altova:current-time-no-TZ() as xs:time XP3.1 XQ3.1
Esta función no toma ningún argumento. Quita la parte de zona horaria de current-time() (que es la hora actual según el reloj del sistema) y devuelve un valor de tipo xs:time.

⊟ *Ejemplos*

Si la hora actual es **14:00:00+01:00**:

- **altova:current-time-no-TZ()** devuelve 14:00:00

▼ date-no-TZ [altova:]

altova:date-no-TZ(InputDate *as xs:date***)** como **xs:date** XP3.1 XQ3.1
Esta función toma un argumento xs:date, del que elimina la parte timezone y devuelve un valor xs:date. Observe que la fecha permanece intacta.

⊞ *Ejemplos*

- **altova:date-no-TZ**(xs:date("2014-01-15+01:00")) devuelve 2014-01-15

▼ dateTime-no-TZ [altova:]

altova:dateTime-no-TZ(InputDateTime *as xs:dateTime***)** como **xs:dateTime** XP3.1 XQ3.1
Esta función toma un argumento xs:dateTime, del que elimina la parte timezone, y devuelve un valor xs:dateTime. Observe que tanto la fecha como la hora permanecen intactas.

⊞ *Ejemplos*

- **altova:dateTime-no-TZ**(xs:date("2014-01-15T14:00:00+01:00")) devuelve 2014-01-15T14:00:00

▼ time-no-TZ [altova:]

altova:time-no-TZ(HoraEntrada *como xs:time***)** como **xs:time** XP3.1 XQ3.1
Esta función toma un argumento `xs:time`, quita la parte de la zona horaria y devuelve un valor `xs:time`. Tenga en cuenta que la hora no se modifica.

⊟ *Ejemplos*

- **altova:time-no-TZ**`(xs:time("14:00:00+01:00"))` devuelve `14:00:00`

[Subir 420]

Devolver el número de días, horas, minutos y segundos de duraciones XP3.1 XQ3.1

Estas funciones devuelven el número de días en un mes y el número de horas, minutos y segundos de las duraciones correspondientes.

▼ days-in-month [altova:]

altova:days-in-month(Year *as xs:integer*, **Month** *as xs:integer***)** como **xs:integer** XP3.1 XQ3.1
Devuelve el número de días en el mes indicado. El mes se indica con los argumentos `Year` y `Month`.

⊞ *Ejemplos*

- **altova:days-in-month**`(2018, 10)` devuelve `31`
- **altova:days-in-month**`(2018, 2)` devuelve `28`
- **altova:days-in-month**`(2020, 2)` devuelve `29`

▼ hours-from-dayTimeDuration-accumulated

altova:hours-from-dayTimeDuration-accumulated(DayAndTime *como xs:duration***)** como **xs:integer** XP3.1 XQ3.1
Devuelve el número total de horas de la duración enviada por el argumento `DayAndTime` (que es de tipo `xs:duration`). Las horas de los componentes `Day` y `Time` se agregan juntos para dar como resultado un número entero. Una hora nueva son 60 minutos enteros. Las duraciones negativas dan como resultado un valor de hora negativo.

⊞ *Ejemplos*

- **altova:hours-from-dayTimeDuration-accumulated**`(xs:duration("P5D")` devuelve `120`, que es el número total de horas en 5 días.
- **altova:hours-from-dayTimeDuration-accumulated**`(xs:duration("P5DT2H")` devuelve `122`, que es el número total de horas en 5 días más 2 horas.
- **altova:hours-from-dayTimeDuration-accumulated**`(xs:duration("P5DT2H60M")` devuelve `123`, que es el número total de horas en 5 días más 2 horas y 60 mins.
- **altova:hours-from-dayTimeDuration-accumulated**`(xs:duration("P5DT2H119M")` devuelve `123`, que es el número total de horas en 5 días más 2 horas y 119 mins.
- **altova:hours-from-dayTimeDuration-accumulated**`(xs:duration("P5DT2H120M")` devuelve `124`, que es el número total de horas en 5 días más 2 horas y 120 mins.
- **altova:hours-from-dayTimeDuration-accumulated**`(xs:duration("-P5DT2H")` devuelve `-122`

 © 2017-2023 Altova GmbH

▼ minutes-from-dayTimeDuration-accumulated

altova:minutes-from-dayTimeDuration-accumulated(DayAndTime *como xs:duration*) como **xs:integer** XP3.1 XQ3.1

Devuelve el número total de minutos de la duración enviada por el argumento `DayAndTime` (que es de tipo `xs:duration`). Los minutos de los componentes `Day` y `Time` se agregan juntos para dar como resultado un número entero. Las duraciones negativas dan como resultado un valor de minutos negativo.

⊞ *Ejemplos*

- **altova:minutes-from-dayTimeDuration-accumulated**(xs:duration("PT60M") devuelve 60
- **altova:minutes-from-dayTimeDuration-accumulated**(xs:duration("PT1H") devuelve 60, que es el número total de minutos en 1 hora.
- **altova:minutes-from-dayTimeDuration-accumulated**(xs:duration("PT1H40M") devuelve 100
- **altova:minutes-from-dayTimeDuration-accumulated**(xs:duration("P1D") devuelve 1440, que es el número total de minutos en 1 día.
- **altova:minutes-from-dayTimeDuration-accumulated**(xs:duration("-P1DT60M") devuelve -1500

▼ seconds-from-dayTimeDuration-accumulated

altova:seconds-from-dayTimeDuration-accumulated(DayAndTime *como xs:duration*) como **xs:integer** XP3.1 XQ3.1

Devuelve el número total de segundos de la duración enviada por el argumento `DayAndTime` (que es de tipo `xs:duration`). Los segundos de los componentes Day y Time se agregan juntos para dar como resultado un número entero. Las duraciones negativas dan como resultado un valor de segundos negativo.

⊞ *Ejemplos*

- **altova:seconds-from-dayTimeDuration-accumulated**(xs:duration("PT1M") devuelve 60, que es el número total de segundos en 1 minuto.
- **altova:seconds-from-dayTimeDuration-accumulated**(xs:duration("PT1H") devuelve 3600, que es el número total de segundos en 1 hora.
- **altova:seconds-from-dayTimeDuration-accumulated**(xs:duration("PT1H2M") devuelve 3720
- **altova:seconds-from-dayTimeDuration-accumulated**(xs:duration("P1D") devuelve 86400, que es el número total de segundos en 1 día.
- **altova:seconds-from-dayTimeDuration-accumulated**(xs:duration("-P1DT1M") devuelve -86460

Obtener el día de la semana de **xs:dateTime** o **xs:date** XP3.1 XQ3.1

Estas funciones obtienen el día de la semana (como entero) de `xs:dateTime` o `xs:date`. Los días de la semana se numeran del `1` al `7` (usando el formato EE UU, es decir `Domingo =1`). En el formato europeo la semana empieza el lunes (es decir, `Lunes=1`). Para establecer el formato EE UU (`Domingo=1`) use el entero `0` allí donde se acepte un entero para indicar el formato.

▼ weekday-from-dateTime [altova:]

altova:weekday-from-dateTime(DateTime *como xs:dateTime*) **como xs:integer** XP3.1 XQ3.1
Toma una fecha como único argumento y devuelve el día de la semana de la fecha dada como número entero. Los días de la semana se numeran del 1 al 7 empezando por `Domingo=1`. Si necesita usar el formato europeo (donde `Lunes=1`), utilice la otra firma de esta función (*ver más abajo*).

Ejemplos

- **altova:weekday-from-dateTime**(xs:dateTime("2014-02-03T09:00:00")) devuelve 2, lo cual significa "Lunes"..

altova:weekday-from-dateTime(DateTime *como xs:dateTime*, **Formato** *como xs:integer*) **como xs:integer** XP3.1 XQ3.1
Toma una fecha como primer argumento y devuelve el día de la semana de la fecha dada como número entero. Si el segundo argumento (número entero) es `0`, entonces los días de la semana se numeran del `1` al `7` empezando por `Domingo=1`. Si el segundo argumento es un entero distinto de `0`, entonces `Lunes=1`. Si falta el segundo argumento, la función se lee como en la firma anterior (*ver más arriba*).

Ejemplos

- **altova:weekday-from-dateTime**(xs:dateTime("2014-02-03T09:00:00"), 1) devuelve 1, lo cual significa "Lunes"
- **altova:weekday-from-dateTime**(xs:dateTime("2014-02-03T09:00:00"), 4) devuelve 1, lo cual significa "Lunes"
- **altova:weekday-from-dateTime**(xs:dateTime("2014-02-03T09:00:00"), 0) devuelve 2, lo cual significa "Lunes"

weekday-from-date [altova:]

altova:weekday-from-date(Date *como xs:date*) **como xs:integer** XP3.1 XQ3.1
Toma una fecha como único argumento y devuelve el día de la semana de la fecha dada como número entero. Los días de la semana se numeran del 1 al 7 empezando por `Domingo=1`. Si necesita usar el formato europeo (donde `Lunes=1`), utilice la otra firma de esta función (*ver más abajo*).

Ejemplos

- **altova:weekday-from-date**(xs:date("2014-02-03+01:00")) devuelve 2, lo cual significa lunes.

altova:weekday-from-date(Date *como xs:date*, **Formato** *como xs:integer*) **como xs:integer** XP3.1 XQ3.1
Toma una fecha como primer argumento y devuelve el día de la semana de la fecha dada como número entero. Si el segundo argumento (`Formato`) es `0`, entonces los días de la semana se numeran del `1` al `7` empezando por `Domingo=1`. Si el segundo argumento es un entero distinto de `0`, entonces `Lunes=1`. Si falta el segundo argumento, la función se lee como en la firma anterior (*ver más arriba*).

Ejemplos

- **altova:weekday-from-date**(xs:date("2014-02-03"), 1) devuelve 1, lo cual significa "Lunes"
- **altova:weekday-from-date**(xs:date("2014-02-03"), 4) devuelve 1, lo cual significa "Lunes"
- **altova:weekday-from-date**(xs:date("2014-02-03"), 0) devuelve 2, lo cual significa "Lunes".

[Subir 420]

 © 2017-2023 Altova GmbH

Devolver el número de semana de xs:dateTime o xs:date XP2 XQ1 XP3.1 XQ3.1

Estas funciones devuelven el número de semana (como número entero) de `xs:dateTime` o `xs:date`. El número de la semana está disponible en el formato de calendario estadounidense, europeo e islámico. La razón de que los números de semana difieran en cada uno de estos calendarios es que en cada uno de ellos se considera un día diferente para el inicio de la semana (p. ej. en el formato estadounidense el primer día de la semana es el domingo).

weeknumber-from-date [altova:]

altova:weeknumber-from-date(Fecha *como xs:date*, **Calendario** *como xs:integer*) como **xs:integer** XP2 XQ1 XP3.1 XQ3.1

Devuelve como número entero el número de semana del argumento **Fecha** dado. El segundo argumento (**Calendario**) indica el sistema de calendario que se debe seguir.
Estos son los valores permitidos para el argumento **Calendario**:

- **0 = Calendario estadounidense** (la semana comienza el domingo)
- **1 = Calendario estándar ISO o europeo** (la semana comienza el lunes)
- **2 = Calendario islámico** (la semana comienza el sábado)

El valor predeterminado es `0`.

Ejemplos

- `altova:weeknumber-from-date(xs:date("2014-03-23"), 0)` devuelve `13`
- `altova:weeknumber-from-date(xs:date("2014-03-23"), 1)` devuelve `12`
- `altova:weeknumber-from-date(xs:date("2014-03-23"), 2)` devuelve `13`
- `altova:weeknumber-from-date(xs:date("2014-03-23") )` devuelve `13`

El día de la fecha de los ejemplos anteriores (`2014-03-23`) es un domingo. Por tanto, en este caso, el calendario estadounidense y el islámico van una semana por delante del calendario europeo.

weeknumber-from-dateTime [altova:]

altova:weeknumber-from-dateTime(FechaHora *como xs:dateTime*, **Calendario** *como xs:integer*) como **xs:integer** XP2 XQ1 XP3.1 XQ3.1

Devuelve como entero el día de la semana del argumento **FechaHora** dado. El segundo argumento (**Calendario**) indica el sistema de calendario que se debe seguir.
Estos son los valores permitidos para el argumento **Calendario**:

- **0 = Calendario estadounidense** (la semana comienza el domingo)
- **1 = Calendario estándar ISO o europeo** (la semana comienza el lunes)
- **2 = Calendario islámico** (la semana comienza el sábado)

El valor predeterminado es `0`.

Ejemplos

- `altova:weeknumber-from-dateTime(xs:dateTime("2014-03-23T00:00:00"), 0)` devuelve `13`
- `altova:weeknumber-from-dateTime(xs:dateTime("2014-03-23T00:00:00"), 1)` devuelve `12`
- `altova:weeknumber-from-dateTime(xs:dateTime("2014-03-23T00:00:00"), 2)` devuelve `13`
- `altova:weeknumber-from-dateTime(xs:dateTime("2014-03-23T00:00:00") )` devuelve `13`

El día de `dateTime` de los ejemplos anteriores (`2014-03-23T00:00:00`) es un domingo. Por tanto, en este caso, el calendario estadounidense y el islámico van una semana por delante del calendario europeo.

[Subir 420]

Generar tipos de datos de fecha, hora y duración a partir de sus componentes léxicos XP3.1 XQ3.1

Estas funciones toman los componentes léxicos de los tipos de datos `xs:date`, `xs:time` y `xs:duration` como argumentos de entrada y los combinan para generar el tipo de datos correspondiente.

build-date [altova:]

altova:build-date(Año *as xs:integer*, **Mes** *as xs:integer*, **Fecha** *as xs:integer*) como **xs:date** XP3.1 XQ3.1

Los argumentos son el año, el mes y la fecha respectivamente. Se combinan para generar un valor de tipo `xs:date`. Los valores de los enteros deben estar en el intervalo de esa fecha en particular. Por ejemplo, el segundo argumento (para el mes) no puede ser mayor que `12`.

Ejemplos

- **altova:build-date**(2014, 2, 03) devuelve 2014-02-03

build-time [altova:]

altova:build-time(Horas *as xs:integer*, **Minutos** *as xs:integer*, **Segundos** *as xs:integer*) **como xs:time** XP3.1 XQ3.1

El primer, segundo y tercer argumentos son la hora (`0` - `23`), los minutos (`0` - `59`) y los segundos (`0` - `59`) respectivamente. Se combinan para generar un valor de tipo `xs:time`. Los valores de los enteros deben estar dentro del intervalo correcto de esa parte de tiempo concreta. Por ejemplo, el segundo argumento (`Minutos`) no puede ser mayor que `59`. Para añadir la parte de uso horario al valor, use la firma que aparece más abajo.

Ejemplos

- **altova:build-time**(23, 4, 57) devuelve 23:04:57

altova:build-time(Horas *como xs:integer*, **Minutos** *como xs:integer*, **Segundos** *as xs:integer*, **TimeZone** *como xs:string*) **como xs:time** XP3.1 XQ3.1

El primer, segundo y tercer argumentos son la hora (`0` - `23`), los minutos (`0` - `59`) y los segundos (`0` - `59`) respectivamente. El cuarto argumento es una cadena de texto que indica la parte del valor de la zona horaria. Este cuarto argumento se combina para generar un valor de tipo `xs:time`. Los valores de los enteros deben estar dentro del intervalo correcto de esa parte de tiempo concreta. Por ejemplo, el segundo argumento (`Minutos`) no puede ser mayor que `59`.

Ejemplos

- **altova:build-time**(23, 4, 57, '+1') devuelve 23:04:57+01:00

 © 2017-2023 Altova GmbH

▼ build-duration [altova:]

altova:build-duration(Años *as xs:integer*, **Meses** *as xs:integer*) como **xs:yearMonthDuration** XP3.1 XQ3.1
Toma dos argumentos para generar un valor de tipo `xs:yearMonthDuration`. El primer argumento da la parte `Years` del valor de duración, mientras que el segundo da la parte `Months`. Si el segundo (`Months`) es mayor o igual que `12`, el entero se divide por 12. El cociente se añade al primer argumento para aportar la parte `Years` del valor de duración, mientras que el resto (de la división) da la parte `Months`. Para generar una duración de tipo `xs:dayTimeDuration`, consulte la firma siguiente.

▣ *Ejemplos*

- **altova:build-duration**(2, 10) devuelve `P2Y10M`
- **altova:build-duration**(14, 27) devuelve `P16Y3M`
- **altova:build-duration**(2, 24) devuelve `P4Y`

altova:build-duration(Días *as xs:integer*, **Horas** *as xs:integer*, **Minutos** *as xs:integer*, **Segundos** *as xs:integer*) como **xs:dayTimeDuration** XP3.1 XQ3.1
Toma cuatro argumentos y los combina para generar un valor de tipo `xs:dayTimeDuration`. El primer argumento da la parte `Days` del valor de duración, el segundo, el tercero y el cuarto dan las partes `Hours`, `Minutes` y `Seconds` respectivamente. Los tres argumentos de tiempo se convierten a un valor equivalente en cuanto a la unidad mayor siguiente y el resultado se utiliza para calcular el valor total de la duración. Por ejemplo, 72 segundos se convierte en `1M+12S` (1 minuto y 12 segundos) y este valor se usa para calcular el valor total de la duración. Para generar una duración de tipo `xs:yearMonthDuration`, consulte la firma anterior.

▣ *Ejemplos*

- **altova:build-duration**(2, 10, 3, 56) devuelve `P2DT10H3M56S`
- **altova:build-duration**(1, 0, 100, 0) devuelve `P1DT1H40M`
- **altova:build-duration**(1, 0, 0, 3600) devuelve `P1DT1H`

[Subir 420]

Construir tipos de datos date, dateTime y time a partir de una cadena de entrada XP2 XQ1 XP3.1 XQ3.1

Estas funciones toman cadenas como argumentos y construyen tipos de datos `xs:date`, `xs:dateTime` o `xs:time`. La cadena de entrada se analiza para los componentes del tipo de datos en función del argumento patrón dado.

▼ parse-date [altova:]

altova:parse-date(Fecha *como xs:string*, **PatrónFecha** *como xs:string*) como **xs:date** XP2 XQ1 XP3.1 XQ3.1
Devuelve la cadena de entrada **Fecha** como valor **xs:date**. El segundo argumento (**PatrónFecha**) indica el patrón (secuencia de componentes) de la cadena de entrada. El argumento **PatrónFecha** se describe con los especificadores que aparecen a continuación y con cualquier separador de componentes (consulte los ejemplos más abajo).

D	Día
M	Mes

Y	Año

El patrón **PatrónFecha** debe coincidir con el patrón de **Fecha**. Como el resultado es de tipo **xs:date**, el resultado siempre tendrá el formato léxico **YYYY-MM-DD**.

Ejemplos

- **altova:parse-date**(xs:string("09-12-2014"), "[D]-[M]-[Y]") devuelve 2014-12-09
- **altova:parse-date**(xs:string("09-12-2014"), "[M]-[D]-[Y]") devuelve 2014-09-12
- **altova:parse-date**("06/03/2014", "[M]/[D]/[Y]") devuelve 2014-06-03
- **altova:parse-date**("06 03 2014", "[M] [D] [Y]") devuelve 2014-06-03
- **altova:parse-date**("6 3 2014", "[M] [D] [Y]") devuelve 2014-06-03

parse-dateTime [altova:]

altova:parse-dateTime(FechaHora *como xs:string*, **PatrónFechaHora** *como xs:string***)** como **xs:dateTime** XP2 XQ1 XP3.1 XQ3.1

Devuelve la cadena de entrada **FechaHora** como valor **xs:dateTime**. El segundo argumento (**PatrónFechaHora**) indica el patrón (secuencia de componentes) de la cadena de entrada. El argumento **PatrónFechaHora** se describe con los especificadores que aparecen a continuación y con cualquier separador de componentes (consulte los ejemplos más abajo).

D	Día
M	Mes
Y	Año
H	Hora
m	minutos
s	segundos

El patrón **PatrónFechaHora** debe coincidir con el patrón de **FechaHora**. Como el resultado es de tipo **xs:dateTime**, el resultado siempre tendrá el formato léxico **YYYY-MM-DDTHH:mm:ss**.

Ejemplos

- **altova:parse-dateTime**(xs:string("09-12-2014 13:56:24"), "[M]-[D]-[Y] [H]:[m]:[s]") devuelve 2014-09-12T13:56:24
- **altova:parse-dateTime**("time=13:56:24; date=09-12-2014", "time=[H]:[m]:[s]; date=[D]-[M]-[Y]") devuelve 2014-12-09T13:56:24

parse-time [altova:]

altova:parse-time(Hora *como xs:string*, **PatrónHora** *como xs:string***)** como **xs:time** XP2 XQ1 XP3.1 XQ3.1

Devuelve la cadena de entrada **Hora** como valor **xs:time**. El segundo argumento (**PatrónHora**) indica el patrón (secuencia de componentes) de la cadena de entrada. El argumento **PatrónHora** se describe con los especificadores que aparecen a continuación y con cualquier separador de componentes (consulte los ejemplos más abajo).

© 2017-2023 Altova GmbH

H	Hora
m	minutos
s	segundos

El patrón **PatrónHora** debe coincidir con el patrón de **Hora**. Como el resultado es de tipo **xs:time**, el resultado siempre tendrá el formato léxico **HH:mm:ss**.

Ejemplos

- **altova:parse-time**(xs:string("13:56:24"), "[H]:[m]:[s]") devuelve 13:56:24
- **altova:parse-time**("13-56-24", "[H]-[m]") devuelve 13:56:00
- **altova:parse-time**("time=13h56m24s", "time=[H]h[m]m[s]s") devuelve 13:56:24
- **altova:parse-time**("time=24s56m13h", "time=[s]s[m]m[H]h") devuelve 13:56:24

[Subir 420]

Funciones para calcular la edad XP3.1 XQ3.1

Estas funciones devuelven la edad que se calcula obteniendo la diferencia (i) entre la fecha del argumento de entrada y la fecha actual o (ii) entre las fechas de los dos argumentos de entrada. La función **age** devuelve la edad en años, mientras que la función **age-details** devuelve la edad en forma de una secuencia de tres enteros (años, meses y días).

age [altova:]

altova:age(FechaInicio *as xs:date***)** como **xs:integer** XP3.1 XQ3.1
Devuelve un entero que es la edad *en años* de algún objeto, contando a partir de la fecha de inicio dada como argumento y hasta la fecha actual (tomada del reloj del sistema). Si el argumento de entrada es un año o más después que la fecha actual, el valor devuelto será negativo.

Ejemplos

Si la fecha actual es **2014-01-15**:

- **altova:age**(xs:date("2013-01-15")) devuelve 1
- **altova:age**(xs:date("2013-01-16")) devuelve 0
- **altova:age**(xs:date("2015-01-15")) devuelve -1
- **altova:age**(xs:date("2015-01-14")) devuelve 0

altova:age(FechaInicio *as xs:date*, **FechaFinal** *as xs:date***)** como **xs:integer** XP3.1 XQ3.1
Devuelve un entero que es la edad *en años* de algún objeto, contando a partir de la fecha de inicio dada como primer argumento y hasta la fecha dada como segundo argumento. El valor devuelto será negativo si el primer argumento es un año o más después que el segundo argumento.

Ejemplos

- **altova:age**(xs:date("2000-01-15"), xs:date("2010-01-15")) devuelve 10
- **altova:age**(xs:date("2000-01-15"), current-date()) devuelve 14 si la fecha actual es

2014-01-15
- **altova:age**(xs:date("2014-01-15"), xs:date("2010-01-15")) devuelve -4

▼ age-details [altova:]

altova:age-details(FechaEntrada *as xs:date***)** como (**xs:integer)*** XP3.1 XQ3.1
Devuelve tres enteros que son los años, meses y días respectivamente que hay entre la fecha dada como argumento y la fecha actual (tomada del reloj del sistema). La suma del valor devuelto nos da el tiempo total transcurrido entre ambas fechas (entre la fecha dada y la fecha actual). La fecha de entrada puede tener un valor anterior o posterior a la fecha actual, pero esto no se indica en el valor devuelto por medio de un signo negativo o positivo. El valor devuelto siempre es positivo.

▣ *Ejemplos*

Si la fecha actual es **2014-01-15**:

- **altova:age-details**(xs:date("2014-01-16")) devuelve (0 0 1)
- **altova:age-details**(xs:date("2014-01-14")) devuelve (0 0 1)
- **altova:age-details**(xs:date("2013-01-16")) devuelve (1 0 1)
- **altova:age-details**(current-date()) devuelve (0 0 0

altova:age-details(Fecha1 *as xs:date*, **Fecha2** *as xs:date***)** como (**xs:integer)*** XP3.1 XQ3.1
Devuelve tres enteros que son los años, meses y días que hay entre las dos fechas dadas por los argumentos. La suma del valor devuelto nos da el tiempo total transcurrido entre las dos fechas de entrada. Da igual cuál de las dos fechas se da como primer argumento, la más antigua o la más reciente. El valor devuelto no indica si la fecha de entrada es anterior o posterior a la fecha actual. Es decir, el valor devuelto siempre es positivo.

▣ *Ejemplos*

- **altova:age-details**(xs:date("2014-01-16"), xs:date("2014-01-15")) devuelve (0 0 1)
- **altova:age-details**(xs:date("2014-01-15"), xs:date("2014-01-16")) devuelve (0 0 1)

[Subir 420]

Funciones para calcular el tiempo Unix XP3.1 XQ3.1

El tiempo Unix es una medida de tiempo que se usa en sistemas Unix. Se define como la cantidad de segundos transcurridos desde las 00:00:00 UTC del 1 de enero de 1970. Estas funciones convierten valores **xs:dateTime** en tiempo Unix y viceversa.

▼ dateTime-from-epoch [altova:]

altova:dateTime-from-epoch(Epoch *como xs:decimal* **como xs:dateTime** XP3.1 XQ3.1
El tiempo Unix es una medida de tiempo que se usa en sistemas Unix. Se define como la cantidad de segundos transcurridos desde las 00:00:00 UTC del 1 de enero de 1970. La función **dateTime-from-epoch** devuelve el equivalente en **xs:dateTime** de un instante de tiempo Unix, lo ajusta a la zona horaria local e incluye la información de esa zona horaria en el resultado.

La función toma un argumento xs:decimal y devuelve un valor xs:dateTime que incluye una parte **TZ**,

 © 2017-2023 Altova GmbH

que indica la zona horaria. Para obtener el resultado se calcula el equivalente en **dateTime** UTC del instante de tiempo Unix y se añade a la zona horaria local (que se obtiene del reloj del sistema). Por ejemplo, si la función se ejecuta en un equipo cuya configuración sitúa en una zona horaria de +01:00 (con respecto a UTC), una vez se ha calculado el equivalente en **dateTime** se le añade una hora al resultado. La información de la zona horaria, que es una parte léxica opcional del resultado de **xs:dateTime**, también se incluye en el resultado **dateTime**. Compare este resultado con el de **dateTime-from-epoch-no-TZ** y consulte la función **epoch-from-dateTime**.

Ejemplos

La zona horaria local de los ejemplos siguientes es UTC +01:00. En consecuencia, el equivalente en **dateTime** UTC del instante de tiempo Unix indicado aumentará en una hora. La zona horaria se indica en el resultado.

- **altova:dateTime-from-epoch**(34) devuelve 1970-01-01T01:00:34+01:00
- **altova:dateTime-from-epoch**(62) devuelve 1970-01-01T01:01:02+01:00

dateTime-from-epoch-no-TZ [altova:]

altova:dateTime-from-epoch-no-TZ(Epoch *como xs:decimal* **como xs:dateTime** XP3.1 XQ3.1
El tiempo Unix es una medida de tiempo que se usa en sistemas Unix. Se define como la cantidad de segundos transcurridos desde las 00:00:00 UTC del 1 de enero de 1970. La función **dateTime-from-epoch-no-TZ** devuelve el equivalente en **xs:dateTime** de un instante de tiempo Unix y lo ajusta a la zona horaria local pero no incluye la información de esa zona horaria en el resultado.

La función toma un argumento xs:decimal y devuelve un valor xs:dateTime que no incluye la parte **TZ**, que indica la zona horaria. Para obtener el resultado se calcula el equivalente en **dateTime** UTC del instante de tiempo Unix y se añade a la zona horaria local (que se obtiene del reloj del sistema). Por ejemplo, si la función se ejecuta en un equipo cuya configuración sitúa en una zona horaria de +01:00 (con respecto a UTC), una vez se ha calculado el equivalente en **dateTime** se le añade una hora al resultado. La información de la zona horaria, que es una parte léxica opcional del resultado de **xs:dateTime**, no se incluye en el resultado **dateTime**. Compare este resultado con el de **dateTime-from-epoch** y consulte la función **epoch-from-dateTime**.

Ejemplos

La zona horaria local de los ejemplos siguientes es UTC +01:00. En consecuencia, el equivalente en **dateTime** UTC del instante de tiempo Unix indicado aumentará en una hora. La zona horaria no se indica en el resultado.

- **altova:dateTime-from-epoch**(34) devuelve 1970-01-01T01:00:34
- **altova:dateTime-from-epoch**(62) devuelve 1970-01-01T01:01:02

epoch-from-dateTime [altova:]

altova:epoch-from-dateTime(dateTimeValue *como xs:dateTime*) **como xs:decimal** XP3.1 XQ3.1
El tiempo Unix es una medida de tiempo que se usa en sistemas Unix. Se define como la cantidad de segundos transcurridos desde las 00:00:00 UTC del 1 de enero de 1970. La función **epoch-from-dateTime** devuelve el equivalente en tiempo Unix del valor **xs:dateTime** que se indica en el argumento de la función. Tenga en cuenta que puede que deba generar de forma explícita el valor **xs:dateTime**. El valor

xs:dateTime puede o no contener la parte opcional **TZ**, que indica la zona horaria.

Tanto si se indica la parte de la zona horaria como parte del argumento como si no, la diferencia que esta indica se obtiene del reloj del sistema y se resta al argumento **dateTimeValue** indicado. El resultado es el tiempo UTC a partir del cual se calcula el equivalente en tiempo Unix. Por ejemplo, si la función se ejecuta en un equipo cuya configuración sitúa en una zona horaria de +01:00 (con respecto a UTC), se resta una hora al valor **dateTimeValue** indicado antes de calcular el valor en tiempo Unix. Consulte también la función **dateTime-from-epoch**.

Ejemplos

La zona horaria local de los ejemplos siguientes es UTC +01:00. En consecuencia, se le restará una hora al valor **dateTime** indicado antes de calcular el tiempo Unix.

- **altova:epoch-from-dateTime**(xs:dateTime("1970-01-01T01:00:34+01:00")) devuelve 34
- **altova:epoch-from-dateTime**(xs:dateTime("1970-01-01T01:00:34")) devuelve 34
- **altova:epoch-from-dateTime**(xs:dateTime("2021-04-01T11:22:33")) devuelve 1617272553

[Subir 420]

10.2.1.3 Funciones XPath/XQuery: de geoubicación

Las funciones de extensión XPath/XQuery de geoubicación son compatibles con la versión actual de RaptorXML Server y se pueden utilizar en (i) expresiones XPath en contextos XSLT o (ii) expresiones XQuery en documentos XQuery.

Nota sobre el nombre de las funciones y lenguajes

Puede utilizar todas las funciones de extensión de Altova en sus expresiones XPath/XQuery. Con ellas conseguirá funciones adicionales no disponibles en la biblioteca de funciones estándar de XPath, XQuery y XSLT. Las funciones de extensión de Altova están en el **espacio de nombres** **http://www.altova.com/xslt-extensions** y en esta sección se presentan con el prefijo, que se supone estará enlazado al espacio de nombres señalado. Tenga en cuenta que en futuras versiones del producto algunas funciones pueden dejar de ser compatibles o su comportamiento puede cambiar. Por tanto, consulte siempre la documentación del producto para conocer el funcionamiento de estas funciones en cada versión del producto.

Funciones XPath (en expresiones XPath en XSLT):	XP1 XP2 XP3.1.1
Funciones XSLT (en expresiones XPath en XSLT):	XSLT1 XSLT2 XSLT3
Funciones XQuery (en expresiones XQuery en XQuery):	XQ1 XQ3.1

format-geolocation [altova:]

altova:format-geolocation(Latitude *como xs:decimal*, **Longitude** *como xs:decimal*,

GeolocationOutputStringFormat *como xs:integer*) como **xs:string** **XP3.1** **XQ3.1**
Toma la latitud y la longitud como los dos primeros argumentos y da como resultado la geoubicación como cadena. El tercer argumento, **GeolocationOutputStringFormat**, es el formato de la cadena de resultado de la geoubicación: usa valores enteros del 1 al 4 para identificar el formato de la cadena de resultado (consulte más abajo "*Formatos de la cadena de resultado geoubicación").* Los valores de latitud oscilan entre `+90` y `-90 (N a S)`. Los valores de longitud oscilan entre +`180` y `-180 (E a O)`.

Nota: la función `image-exif-data` y el atributo de metadatos Exif se pueden usar para suministrar las cadenas de entrada.

- *Ejemplos*
  - **altova:format-geolocation**`(33.33, -22.22, 4)` devuelve el `xs:string "33.33 -22.22"`
  - **altova:format-geolocation**`(33.33, -22.22, 2)` devuelve el `xs:string "33.33N 22.22W"`
  - **altova:format-geolocation**`(-33.33, 22.22, 2)` devuelve el `xs:string "33.33S 22.22E"`
  - **altova:format-geolocation**`(33.33, -22.22, 1)` devuelve el `xs:string "33°19'48.00"S 22° 13'12.00"E"`

- *Formato de las cadenas de salida de las geoubicaciones:*

  A la latitud y longitud suministradas se les aplica un formato de salida de los que se indican más abajo. El formato deseado se identifica con un identificador comprendido entre `1` y `4`. Los valores de latitud pueden estar comprendidos entre `+90` y `-90` (`N` a `S`). Los valores de longitud pueden estar comprendidos entre `+180` y `-180` (`E` a `W`).

1
Grados, minutos y segundos decimales + orientación como sufijo (`N/S, E/W`) **`D°M'S.SS"N/S D°M'S.SS"E/W`** *Ejemplo*: **`33°55'11.11"N 22°44'66.66"W`**

2
Grados decimales + orientación como sufijo (`N/S, E/W`) **`D.DDN/S D.DDE/W`** *Ejemplo*: **`33.33N 22.22W`**

3
Grados, minutos y segundos decimales + prefijo (`+/-`). El signo + para (**`N/E`**) es opcional **`+/-D°M'S.SS" +/-D°M'S.SS"`** *Ejemplo*: **`33°55'11.11" -22°44'66.66"`**

4
Grados decimales + prefijo (`+/-`). El signo + para (**`N/E`**) es opcional **`+/-D.DD +/-D.DD`** *Ejemplo*: **`33.33 -22.22`**

- *Atributo Exif de Altova: Geolocation*

El motor XPath/XQuery de Altova genera el atributo personalizado **Geolocation** a partir de las etiquetas de metadatos Exif estándar. Este atributo es una concatenación de cuatro etiquetas Exif (GPSLatitude, GPSLatitudeRef, GPSLongitude, GPSLongitudeRef) seguidas de unidades:

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(CadenaEntradaGeoubicación *como xs:string*) como **xs:decimal+**
XP3.1 **XQ3.1**
Analiza el argumento CadenaEntradaGeoubicación y devuelve la latitud y la longitud (en ese orden) de la geoubicación en forma de secuencia de dos elementos xs:decimal. Más abajo puede ver en qué formatos se puede suministrar la cadena de entrada de la geoubicación.

Nota: la función image-exif-data 449 y el atributo @Geolocation 449 de los metadatos Exif se pueden utilizar para suministrar la cadena de entrada de la geoubicación (*ver ejemplos*).

- *Ejemplos*
  - **altova:parse-geolocation**("33.33 -22.22") devuelve la secuencia de dos xs:decimals (33.33, 22.22)
  - **altova:parse-geolocation**("48°51'29.6""N 24°17'40.2""") devuelve la secuencia de dos xs:decimals (48.8582222222222, 24.2945)
  - **altova:parse-geolocation**('48°51''29.6"N 24°17''40.2"') devuelve la secuencia de dos xs:decimals (48.8582222222222, 24.2945)
  - **altova:parse-geolocation**(**image-exif-data**(//MisImágenes/Imagen20141130.01)**/@Geolocation**) devuelve una secuencia de dos xs:decimals

- *Formato de las cadenas de entrada de geoubicaciones:*

  La cadena de entrada de la geoubicación debe contener la latitud y la longitud (en ese orden) se paradas por un espacio en blanco. Ambas pueden estar en cualquier formato de los que se indican más abajo y puede combinar formatos distintos. Es decir, la latitud puede estar en un formato y la longitud en otro. Los valores de la latitud deben estar comprendidos entre +90 y -90 (N a S). Los valores de longitud deben estar comprendidos entre +180 y -180 (E a W).

  Nota: si utiliza comillas simples o dobles para delimitar el argumento de la cadena de entrada, esto dará lugar a un conflicto con las comillas simples o dobles que se utilizan, respectivamente, para indicar los valores de los minutos y los segundos. Si esto ocurre, debe añadir caracteres de escape a las comillas utilizadas para los minutos y segundos (esto se hace duplicando las comillas.

  - Grados, minutos y segundos decimales + orientación como sufijo (N/S, E/W)
    D°M'S.SS"N/S D°M'S.SS"W/E
    Ejemplo: **33°55'11.11"N 22°44'55.25"W**

© 2017-2023 Altova GmbH

- Grados, minutos y segundos decimales + prefijo (+/-). El signo + para (**N/E**) es opcional
  +/-D°M'S.SS" +/-D°M'S.SS"
  Ejemplo: **33°55'11.11" -22°44'55.25"**

- Grados y minutos decimales + orientación como sufijo (N/S, E/W)
  D°M.MM'N/S D°M.MM'W/E
  Ejemplo: **33°55.55'N 22°44.44'W**

- Grados y minutos decimales + prefijo (+/-). El signo + para (**N/E**) es opcional
  +/-D°M.MM' +/-D°M.MM'
  Ejemplo: **+33°55.55' -22°44.44'**

- Grados decimales + orientación como sufijo (N/S, E/W)
  D.DDN/S D.DDW/E
  Ejemplo: **33.33N 22.22W**

- Grados decimales + prefijo (+/-). El signo + para (**N/S, E/W**) es opcional
  +/-D.DD +/-D.DD
  Ejemplo: **33.33 -22.22**

Ejemplos de combinación de formatos:
33.33N -22°44'55.25"
33.33 22°44'55.25"W
33.33 22.45

Atributo Exif de Altova: Geolocation

El motor XPath/XQuery de Altova genera el atributo personalizado **Geolocation** a partir de las etiquetas de metadatos Exif estándar. Este atributo es una concatenación de cuatro etiquetas Exif (GPSLatitude, GPSLatitudeRef, GPSLongitude, GPSLongitudeRef) seguidas de unidades:

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(CadenaEntradaGeoubicación-1 *como xs:string*, **CadenaEntradaGeoubicación-2** *como xs:string*) como **xs:decimal** **XP3.1 XQ3.1**
Calcula la distancia en km que existe entre dos geoubicaciones. El formato que puede utilizarse para dar las cadenas de entrada aparece más abajo. Los valores de latitud están comprendidos entre +90 y -90 (N a S). Los valores de longitud están comprendidos entre +180 y -180 (E a W).

Nota: la función image-exif-data 449 y el atributo de metadatos Exif @Geolocation 449 pueden utilizarse para suministrar las cadenas de entrada de geoubicaciones.

Ejemplos

- **altova:geolocation-distance-km**("33.33 -22.22", "48°51'29.6""N 24°17'40.2""")

devuelve el `xs:decimal 4183.08132372392`

Formato de las cadenas de entrada de geoubicaciones:

La cadena de entrada de la geoubicación debe contener la latitud y la longitud (en ese orden) se paradas por un espacio en blanco. Ambas pueden estar en cualquier formato de los que se indican más abajo y puede combinar formatos distintos. Es decir, la latitud puede estar en un formato y la longitud en otro. Los valores de la latitud deben estar comprendidos entre `+90` y `-90` (`N` a `S`). Los valores de longitud deben estar comprendidos entre `+180` y `-180` (`E` a `W`).

Nota: si utiliza comillas simples o dobles para delimitar el argumento de la cadena de entrada, esto dará lugar a un conflicto con las comillas simples o dobles que se utilizan, respectivamente, para indicar los valores de los minutos y los segundos. Si esto ocurre, debe añadir caracteres de escape a las comillas utilizadas para los minutos y segundos (esto se hace duplicando las comillas.

- Grados, minutos y segundos decimales + orientación como sufijo (`N/S`, `E/W`)
  `D°M'S.SS"N/S D°M'S.SS"W/E`
  Ejemplo: **`33°55'11.11"N 22°44'55.25"W`**

- Grados, minutos y segundos decimales + prefijo (`+/-`). El signo + para (**`N/E`**) es opcional
  `+/-D°M'S.SS" +/-D°M'S.SS"`
  Ejemplo: **`33°55'11.11" -22°44'55.25"`**

- Grados y minutos decimales + orientación como sufijo (`N/S`, `E/W`)
  `D°M.MM'N/S D°M.MM'W/E`
  Ejemplo: **`33°55.55'N 22°44.44'W`**

- Grados y minutos decimales + prefijo (`+/-`). El signo + para (**`N/E`**) es opcional
  `+/-D°M.MM' +/-D°M.MM'`
  Ejemplo: **`+33°55.55' -22°44.44'`**

- Grados decimales + orientación como sufijo (`N/S`, `E/W`)
  `D.DDN/S D.DDW/E`
  Ejemplo: **`33.33N 22.22W`**

- Grados decimales + prefijo (`+/-`). El signo + para (**`N/s, E/W`**) es opcional
  `+/-D.DD +/-D.DD`
  Ejemplo: **`33.33 -22.22`**

Ejemplos de combinación de formatos:

```
33.33N  -22°44'55.25"
33.33  22°44'55.25"W
33.33  22.45
```

Atributo Exif de Altova: Geolocation

El motor XPath/XQuery de Altova genera el atributo personalizado **`Geolocation`** a partir de las etiquetas de metadatos Exif estándar. Este atributo es una concatenación de cuatro etiquetas Exif (`GPSLatitude`, `GPSLatitudeRef`, `GPSLongitude`, `GPSLongitudeRef`) seguidas de unidades:

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation

© 2017-2023 Altova GmbH

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(CadenaEntradaGeoubicación-1 *como xs:string*, **CadenaEntradaGeoubicación-2** *como xs:string*) como **xs:decimal** **XP3.1** **XQ3.1**
Calcula la distancia en millas que existe entre dos geoubicaciones. El formato que puede utilizarse para dar las cadenas de entrada aparece más abajo. Los valores de latitud están comprendidos entre +90 y -90 (N a S). Los valores de longitud están comprendidos entre +180 y -180 (E a W).

Nota: la función image-exif-data 449 y el atributo de metadatos Exif @Geolocation 449 pueden utilizarse para suministrar las cadenas de entrada de geoubicaciones.

- *Ejemplos*
  - **altova:geolocation-distance-mi**("33.33 -22.22", "48°51'29.6""N 24°17'40.2""")
    devuelve el xs:decimal 2599.40652340653

- *Formato de las cadenas de entrada de geoubicaciones:*

  La cadena de entrada de la geoubicación debe contener la latitud y la longitud (en ese orden) se paradas por un espacio en blanco. Ambas pueden estar en cualquier formato de los que se indican más abajo y puede combinar formatos distintos. Es decir, la latitud puede estar en un formato y la longitud en otro. Los valores de la latitud deben estar comprendidos entre +90 y -90 (N a S). Los valores de longitud deben estar comprendidos entre +180 y -180 (E a W).

  Nota: si utiliza comillas simples o dobles para delimitar el argumento de la cadena de entrada, esto dará lugar a un conflicto con las comillas simples o dobles que se utilizan, respectivamente, para indicar los valores de los minutos y los segundos. Si esto ocurre, debe añadir caracteres de escape a las comillas utilizadas para los minutos y segundos (esto se hace duplicando las comillas.

  - Grados, minutos y segundos decimales + orientación como sufijo (N/S, E/W)
    D°M'S.SS"N/S D°M'S.SS"W/E
    Ejemplo: **33°55'11.11"N 22°44'55.25"W**

  - Grados, minutos y segundos decimales + prefijo (+/-). El signo + para (**N/E**) es opcional
    +/-D°M'S.SS" +/-D°M'S.SS"
    Ejemplo: **33°55'11.11" -22°44'55.25"**

  - Grados y minutos decimales + orientación como sufijo (N/S, E/W)
    D°M.MM'N/S D°M.MM'W/E
    Ejemplo: **33°55.55'N 22°44.44'W**

  - Grados y minutos decimales + prefijo (+/-). El signo + para (**N/E**) es opcional
    +/-D°M.MM' +/-D°M.MM'
    Ejemplo: **+33°55.55' -22°44.44'**

- Grados decimales + orientación como sufijo (N/S, E/W)
  D.DDN/S D.DDW/E
  Ejemplo: **33.33N 22.22W**

- Grados decimales + prefijo (+/-). El signo + para (**N/S, E/W**) es opcional
  +/-D.DD +/-D.DD
  Ejemplo: **33.33 -22.22**

Ejemplos de combinación de formatos:
33.33N -22°44'55.25"
33.33 22°44'55.25"W
33.33 22.45

Atributo Exif de Altova: Geolocation

El motor XPath/XQuery de Altova genera el atributo personalizado **Geolocation** a partir de las etiquetas de metadatos Exif estándar. Este atributo es una concatenación de cuatro etiquetas Exif (GPSLatitude, GPSLatitudeRef, GPSLongitude, GPSLongitudeRef) seguidas de unidades:

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:geolocations-bounding-rectangle(Geolocations como *xs:sequence*, **GeolocationOutputStringFormat como** *xs:integer*) como **xs:string** XP3.1 XQ3.1
Toma una secuencia de cadenas de texto como primer argumento; cada cadena de esa secuencia es una geoubicación. La función devuelve una secuencia de dos cadenas que son, respectivamente, las coordinadas de geoubicación de la parte superior izquierda e inferior derecha de un rectángulo delimitado que tiene el tamaño exacto para contener las geoubicaciones suministradas en el primer argumento. Más abajo se enumeran los formatos en que se puede dar una cadena de entrada de geoubicación (*véase "Formato de las cadenas de entrada de geoubicaciones"*). Los valores de latitud están comprendidos entre +90 y -90 (N a S). Los valores de longitud están comprendidos entre +180 y -180 (E a W).

El segundo argumento de la función indica el formato de las dos cadenas de geoubicación de la secuencia de salida. El argumento toma un valor entero entre 1 y 4, donde cada valor representa un formato distinto de las cadenas de entrada de geoubicaciones (*véase "Formato de las cadenas de salida de geoubicaciones"*).

Nota: la función image-exif-data 449 y los atributos de metadatos Exif se pueden usar para suministrar las cadenas de entrada.

Ejemplos

- **altova:geolocations-bounding-rectangle**(("48.2143531 16.3707266", "51.50939 -0.11832"), 1) devuelve la secuencia ("51°30'33.804"N 0°7'5.952"W", "48°12'51.67116"N 16°22'14.61576"E")
- **altova:geolocations-bounding-rectangle**(("48.2143531 16.3707266", "51.50939 -0.11832", "42.5584577 -70.8893334"), 4) devuelve la secuencia ("51.50939 -70.8893334", "42.5584577 16.3707266")

 © 2017-2023 Altova GmbH

Formato de las cadenas de entrada de geoubicaciones:

La cadena de entrada de la geoubicación debe contener la latitud y la longitud (en ese orden) se paradas por un espacio en blanco. Ambas pueden estar en cualquier formato de los que se indican más abajo y puede combinar formatos distintos. Es decir, la latitud puede estar en un formato y la longitud en otro. Los valores de la latitud deben estar comprendidos entre `+90` y `-90` (`N` a `S`). Los valores de longitud deben estar comprendidos entre `+180` y `-180` (`E` a `W`).

Nota: si utiliza comillas simples o dobles para delimitar el argumento de la cadena de entrada, esto dará lugar a un conflicto con las comillas simples o dobles que se utilizan, respectivamente, para indicar los valores de los minutos y los segundos. Si esto ocurre, debe añadir caracteres de escape a las comillas utilizadas para los minutos y segundos (esto se hace duplicando las comillas.

- Grados, minutos y segundos decimales + orientación como sufijo (`N/S`, `E/W`)
  `D°M'S.SS"N/S D°M'S.SS"W/E`
  Ejemplo: **`33°55'11.11"N 22°44'55.25"W`**

- Grados, minutos y segundos decimales + prefijo (`+/-`). El signo + para (**`N/E`**) es opcional
  `+/-D°M'S.SS" +/-D°M'S.SS"`
  Ejemplo: **`33°55'11.11" -22°44'55.25"`**

- Grados y minutos decimales + orientación como sufijo (`N/S`, `E/W`)
  `D°M.MM'N/S D°M.MM'W/E`
  Ejemplo: **`33°55.55'N 22°44.44'W`**

- Grados y minutos decimales + prefijo (`+/-`). El signo + para (**`N/E`**) es opcional
  `+/-D°M.MM' +/-D°M.MM'`
  Ejemplo: **`+33°55.55' -22°44.44'`**

- Grados decimales + orientación como sufijo (`N/S`, `E/W`)
  `D.DDN/S D.DDW/E`
  Ejemplo: **`33.33N 22.22W`**

- Grados decimales + prefijo (`+/-`). El signo + para (**`N/S, E/W`**) es opcional
  `+/-D.DD +/-D.DD`
  Ejemplo: **`33.33 -22.22`**

Ejemplos de combinación de formatos:

```
33.33N  -22°44'55.25"
33.33  22°44'55.25"W
33.33  22.45
```

Formato de las cadenas de salida de las geoubicaciones:

A la latitud y longitud suministradas se les aplica un formato de salida de los que se indican más abajo. El formato deseado se identifica con un identificador comprendido entre `1` y `4`. Los valores de latitud pueden estar comprendidos entre `+90` y `-90` (`N` a `S`). Los valores de longitud pueden estar comprendidos entre `+180` y `-180` (`E` a `W`).

1
Grados, minutos y segundos decimales + orientación como sufijo (`N/S, E/W`)

D°M'S.SS"N/S D°M'S.SS"E/W *Ejemplo*: **33°55'11.11"N 22°44'66.66"W**

2
Grados decimales + orientación como sufijo (N/S, E/W) **D.DDN/S D.DDE/W** *Ejemplo*: **33.33N 22.22W**

3
Grados, minutos y segundos decimales + prefijo (+/-). El signo + para (**N/E**) es opcional **+/-D°M'S.SS" +/-D°M'S.SS"** *Ejemplo*: **33°55'11.11" -22°44'66.66"**

4
Grados decimales + prefijo (+/-). El signo + para (**N/E**) es opcional **+/-D.DD +/-D.DD** *Ejemplo*: **33.33 -22.22**

Atributo Exif de Altova: Geolocation

El motor XPath/XQuery de Altova genera el atributo personalizado **Geolocation** a partir de las etiquetas de metadatos Exif estándar. Este atributo es una concatenación de cuatro etiquetas Exif (GPSLatitude, GPSLatitudeRef, GPSLongitude, GPSLongitudeRef) seguidas de unidades:

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(Geoubicación *como xs:string*, **((PuntoDePolígono** *como xs:string*)+)) como **xs:boolean** **XP3.1** **XQ3.1**

Determina si **Geoubicación** (primer argumento) está dentro del área poligonal descrita por los argumentos **PuntoDePolígono**. Si los argumentos PuntoDePolígono no forman una figura cerrada (la figura se cierra cuando el primer y el último punto son el mismo), entonces el primer punto se añade implícitamente como último punto a fin de cerrar la figura. Todos los argumentos (Geoubicación y PuntoDePolígono+) se dan como cadenas de entrada de geoubicación (*formatos permitidos más abajo*). Si el argumento Geoubicación está dentro del área poligonal, entonces la función devuelve true(). De lo contrario, devuelve false(). Los valores de latitud están comprendidos entre +90 y -90 (N a S). Los valores de longitud están comprendidos entre +180 y -180 (E a W).

Nota: la función image-exif-data 449 y el atributo de metadatos Exif @Geolocation 449 pueden utilizarse para suministrar las cadenas de entrada de geoubicaciones.

© 2017-2023 Altova GmbH

Ejemplos

- `altova:geolocation-within-polygon("33 -22", ("58 -32", "-78 -55", "48 24", "58 -32"))` devuelve `true()`
- `altova:geolocation-within-polygon("33 -22", ("58 -32", "-78 -55", "48 24"))` devuelve `true()`
- `altova:geolocation-within-polygon("33 -22", ("58 -32", "-78 -55", "48°51'29.6""N 24°17'40.2""))` devuelve `true()`

Formato de las cadenas de entrada de geoubicaciones:

La cadena de entrada de la geoubicación debe contener la latitud y la longitud (en ese orden) se paradas por un espacio en blanco. Ambas pueden estar en cualquier formato de los que se indican más abajo y puede combinar formatos distintos. Es decir, la latitud puede estar en un formato y la longitud en otro. Los valores de la latitud deben estar comprendidos entre `+90` y `-90` (`N` a `S`). Los valores de longitud deben estar comprendidos entre `+180` y `-180` (`E` a `W`).

Nota: si utiliza comillas simples o dobles para delimitar el argumento de la cadena de entrada, esto dará lugar a un conflicto con las comillas simples o dobles que se utilizan, respectivamente, para indicar los valores de los minutos y los segundos. Si esto ocurre, debe añadir caracteres de escape a las comillas utilizadas para los minutos y segundos (esto se hace duplicando las comillas.

- Grados, minutos y segundos decimales + orientación como sufijo (`N/S`, `E/W`)
  `D°M'S.SS"N/S D°M'S.SS"W/E`
  Ejemplo: **`33°55'11.11"N 22°44'55.25"W`**

- Grados, minutos y segundos decimales + prefijo (`+/-`). El signo + para (**`N/E`**) es opcional
  `+/-D°M'S.SS" +/-D°M'S.SS"`
  Ejemplo: **`33°55'11.11" -22°44'55.25"`**

- Grados y minutos decimales + orientación como sufijo (`N/S`, `E/W`)
  `D°M.MM'N/S D°M.MM'W/E`
  Ejemplo: **`33°55.55'N 22°44.44'W`**

- Grados y minutos decimales + prefijo (`+/-`). El signo + para (**`N/E`**) es opcional
  `+/-D°M.MM' +/-D°M.MM'`
  Ejemplo: **`+33°55.55' -22°44.44'`**

- Grados decimales + orientación como sufijo (`N/S`, `E/W`)
  `D.DDN/S D.DDW/E`
  Ejemplo: **`33.33N 22.22W`**

- Grados decimales + prefijo (`+/-`). El signo + para (**`N/S, E/W`**) es opcional
  `+/-D.DD +/-D.DD`
  Ejemplo: **`33.33 -22.22`**

Ejemplos de combinación de formatos:

```
33.33N  -22°44'55.25"
33.33  22°44'55.25"W
33.33  22.45
```

Atributo Exif de Altova: Geolocation

El motor XPath/XQuery de Altova genera el atributo personalizado **Geolocation** a partir de las etiquetas de metadatos Exif estándar. Este atributo es una concatenación de cuatro etiquetas Exif (GPSLatitude, GPSLatitudeRef, GPSLongitude, GPSLongitudeRef) seguidas de unidades:

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(Geoubicación *como xs:string*, **ÁnguloRectángulo-1** *como xs:string*, **ÁnguloRectángulo-2** *como xs:string*) como **xs:boolean** **XP3.1** **XQ3.1**
Determina si **Geoubicación** (primer argumento) está dentro del rectángulo definido por el segundo y el tercer argumento (**ÁnguloRectángulo-1** y **ÁnguloRectángulo-2**), que indican ángulos opuestos del rectángulo. Todos los argumentos de la función se dan como cadenas de entrada de geoubicación (*formatos permitidos más abajo*). Si el argumento Geoubicación está dentro del rectángulo, entonces la función devuelve true(). De lo contrario, devuelve false(). Los valores de latitud están comprendidos entre +90 y -90 (N a S). Los valores de longitud están comprendidos entre +180 y -180 (E a W).

Nota: la función image-exif-data 449 y el atributo de metadatos Exif @Geolocation 449 pueden utilizarse para suministrar las cadenas de entrada de geoubicaciones.

▣ *Ejemplos*

- **altova:geolocation-within-rectangle("33 -22", "58 -32", "-48 24")** devuelve true()
- **altova:geolocation-within-rectangle("33 -22", "58 -32", "48 24")** devuelve false()
- **altova:geolocation-within-rectangle("33 -22", "58 -32", "48°51'29.6""S 24° 17'40.2""")** devuelve true()

▣ *Formato de las cadenas de entrada de geoubicaciones:*

La cadena de entrada de la geoubicación debe contener la latitud y la longitud (en ese orden) se paradas por un espacio en blanco. Ambas pueden estar en cualquier formato de los que se indican más abajo y puede combinar formatos distintos. Es decir, la latitud puede estar en un formato y la longitud en otro. Los valores de la latitud deben estar comprendidos entre +90 y -90 (N a S). Los valores de longitud deben estar comprendidos entre +180 y -180 (E a W).

Nota: si utiliza comillas simples o dobles para delimitar el argumento de la cadena de entrada, esto dará lugar a un conflicto con las comillas simples o dobles que se utilizan, respectivamente, para indicar los valores de los minutos y los segundos. Si esto ocurre, debe añadir caracteres de escape a las comillas utilizadas para los minutos y segundos (esto se hace duplicando las comillas.

- Grados, minutos y segundos decimales + orientación como sufijo (N/S, E/W)
  D°M'S.SS"N/S D°M'S.SS"W/E
  Ejemplo: **33°55'11.11"N 22°44'55.25"W**

- Grados, minutos y segundos decimales + prefijo (+/-). El signo + para (**N/E**) es opcional

© 2017-2023 Altova GmbH

`+/-D°M'S.SS" +/-D°M'S.SS"`
Ejemplo: `33°55'11.11" -22°44'55.25"`

- Grados y minutos decimales + orientación como sufijo (`N/S`, `E/W`)
  `D°M.MM'N/S D°M.MM'W/E`
  Ejemplo: `33°55.55'N 22°44.44'W`

- Grados y minutos decimales + prefijo (`+/-`). El signo + para (`N/E`) es opcional
  `+/-D°M.MM' +/-D°M.MM'`
  Ejemplo: `+33°55.55' -22°44.44'`

- Grados decimales + orientación como sufijo (`N/S`, `E/W`)
  `D.DDN/S D.DDW/E`
  Ejemplo: `33.33N 22.22W`

- Grados decimales + prefijo (`+/-`). El signo + para (`N/S, E/W`) es opcional
  `+/-D.DD +/-D.DD`
  Ejemplo: `33.33 -22.22`

Ejemplos de combinación de formatos:
`33.33N -22°44'55.25"`
`33.33 22°44'55.25"W`
`33.33 22.45`

- *Atributo Exif de Altova: Geolocation*

  El motor XPath/XQuery de Altova genera el atributo personalizado `Geolocation` a partir de las etiquetas de metadatos Exif estándar. Este atributo es una concatenación de cuatro etiquetas Exif (`GPSLatitude`, `GPSLatitudeRef`, `GPSLongitude`, `GPSLongitudeRef`) seguidas de unidades:

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

[Subir 438]

10.2.1.4 Funciones XPath/XQuery: relacionadas con imágenes

Las funciones de extensión XPath/XQuery para trabajar con imágenes son compatibles con la versión actual de RaptorXML Server y se pueden utilizar en (i) expresiones XPath en contextos XSLT o (ii) expresiones XQuery en documentos XQuery.

Nota sobre el nombre de las funciones y lenguajes

Puede utilizar todas las funciones de extensión de Altova en sus expresiones XPath/XQuery. Con ellas conseguirá funciones adicionales no disponibles en la biblioteca de funciones estándar de XPath, XQuery y

XSLT. Las funciones de extensión de Altova están en el **espacio de nombres** `http://www.altova.com/xslt-extensions` y en esta sección se presentan con el prefijo, que se supone estará enlazado al espacio de nombres señalado. Tenga en cuenta que en futuras versiones del producto algunas funciones pueden dejar de ser compatibles o su comportamiento puede cambiar. Por tanto, consulte siempre la documentación del producto para conocer el funcionamiento de estas funciones en cada versión del producto.

Funciones XPath (en expresiones XPath en XSLT):	XP1 XP2 XP3.1.1
Funciones XSLT (en expresiones XPath en XSLT):	XSLT1 XSLT2 XSLT3
Funciones XQuery (en expresiones XQuery en XQuery):	XQ1 XQ3.1

▼ suggested-image-file-extension [altova:]

altova:suggested-image-file-extension(CadenaBase64 *como string***)** como **string?** XP3.1 XQ3.1
Toma la codificación base64 de un archivo de imagen como argumento y devuelve la extensión de archivo de la imagen registrada en la codificación base64 de la imagen. El valor devuelto es una sugerencia basada en la información sobre el tipo de imagen disponible en la codificación. Si esta información no está disponible, entonces devuelve una cadena vacía. Esta función es muy práctica a la hora de guardar una imagen base64 como archivo y recuperar de forma dinámica una extensión de archivo adecuada.

▣ *Ejemplos*

- **altova:suggested-image-file-extension**(/MisImágenes/TeléfonoMóvil/Imagen20141130.01) devuelve `'jpg'`
- **altova:suggested-image-file-extension**($XML1/Personal/Persona/@photo) devuelve `''`

En los ejemplos anteriores, se da por hecho que los nodos suministrados como argumento de la función contienen una imagen codificada en base64. El primer ejemplo recupera `jpg` como tipo de imagen y como extensión de archivo. En el segundo ejemplo, la codificación base64 dada no ofrece información sobre la extensión del archivo.

▼ image-exif-data [altova:]

altova:image-exif-data(CadenaBinariaBase64 *como string***)** como **element?** XP3.1 XQ3.1
Toma una imagen JPEG codificada en base64 como argumento y devuelve un elemento llamado **Exif** que contiene los metadatos Exif de la imagen. Los metadatos Exif se crean como pares atributo-valor del elemento `Exif`. El nombre de los atributos son las etiquetas de datos Exif encontradas en la codificación base64. La lista de etiquetas Exif aparece más abajo. Si en lo datos Exif hay etiquetas de terceros, estas etiquetas y sus valores también se devuelven en un par atributo-valor. Además de las etiquetas de metadatos Exif estándar (*lista más abajo*), también se generan pares atributo-valor de Altova. Estos atributos Exif de Altova también se enumeran más abajo.

▣ *Ejemplos*

- Para acceder a un atributo, utilice la función de esta manera:
  image-exif-data(//MisImágenes/Imagen20141130.01)**/@GPSLatitude**

 © 2017-2023 Altova GmbH

```
image-exif-data(//MisImágenes/Imagen20141130.01)/@Geolocation
```

- Para acceder a todos los atributos, utilice la función de esta manera:
  ```
  image-exif-data(//MisImágenes/Imagen20141130.01)/@*
  ```
- Para acceder al nombre de todos los atributos, utilice esta expresión:
  ```
  for $i in image-exif-data(//MisImágenes/Imagen20141130.01)/@* return name($i)
  ```
  Esto es muy práctico a la hora de averiguar el nombre de los atributos que devuelve la función.

Atributo Exif de Altova: Geolocation

El motor XPath/XQuery de Altova genera el atributo personalizado **Geolocation** a partir de las etiquetas de metadatos Exif estándar. Este atributo es una concatenación de cuatro etiquetas Exif (GPSLatitude, GPSLatitudeRef, GPSLongitude, GPSLongitudeRef) seguidas de unidades:

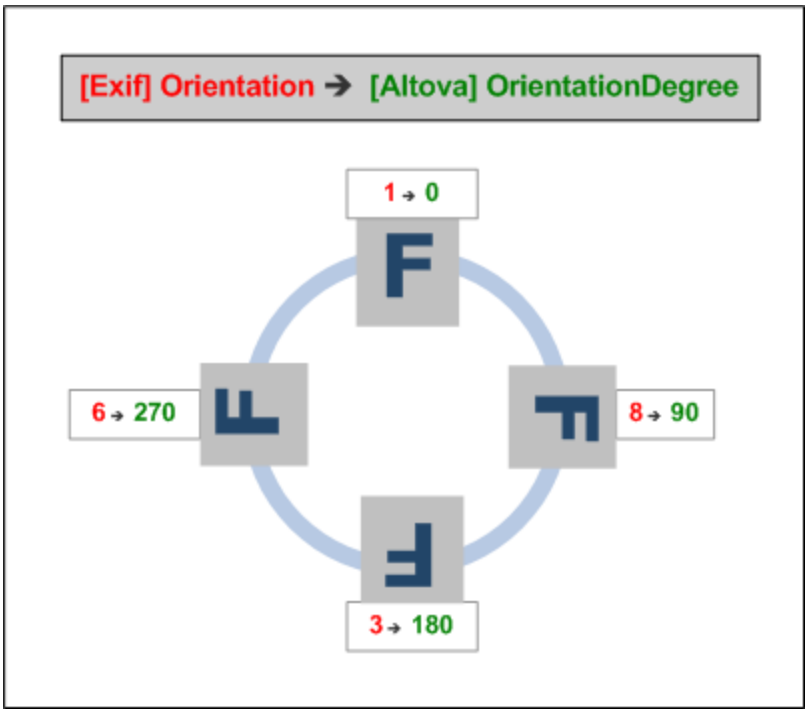
GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

Atributo Exif de Altova: OrientationDegree

El motor XPath/XQuery de Altova genera el atributo personalizado **OrientationDegree** a partir de la etiqueta de metadatos Exif **Orientation**.

Este atributo transforma el valor entero de la etiqueta Exif Orientation (1, 8, 3 o 6) en el correspondiente valor en grados (0, 90, 180, 270), tal y como describe el diagrama más abajo.

Debe tener en cuenta que los valores 2, 4, 5, 7 de Orientation no se pueden traducir. Estas orientaciones se obtienen invirtiendo la imagen 1 en su eje central vertical para obtener la imagen con un valor de 2 e invirtiendo después esta imagen por pasos de 90 grados en el sentido de las agujas del reloj para obtener los valores de 7, 4 y 5, respectivamente.



- *Lista de etiquetas Exif estándar*
  - 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

 © 2017-2023 Altova GmbH

- 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

[Subir 449]

10.2.1.5 Funciones XPath/XQuery: numéricas

Las funciones de extensión numéricas de Altova pueden utilizarse en expresiones XPath y XQuery y ofrecen funciones adicionales para el procesamiento de datos. Estas funciones se pueden usar con los motores **XPath 3.0** y **XQuery 3.0** de Altova. Están disponibles en contextos XPath/XQuery.

Nota sobre el nombre de las funciones y lenguajes

 © 2017-2023 Altova GmbH

Puede utilizar todas las funciones de extensión de Altova en sus expresiones XPath/XQuery. Con ellas conseguirá funciones adicionales no disponibles en la biblioteca de funciones estándar de XPath, XQuery y XSLT. Las funciones de extensión de Altova están en el **espacio de nombres** `http://www.altova.com/xslt-extensions` y en esta sección se presentan con el prefijo, que se supone estará enlazado al espacio de nombres señalado. Tenga en cuenta que en futuras versiones del producto algunas funciones pueden dejar de ser compatibles o su comportamiento puede cambiar. Por tanto, consulte siempre la documentación del producto para conocer el funcionamiento de estas funciones en cada versión del producto.

Funciones XPath (en expresiones XPath en XSLT):	XP1 XP2 XP3.1.1
Funciones XSLT (en expresiones XPath en XSLT):	XSLT1 XSLT2 XSLT3
Funciones XQuery (en expresiones XQuery en XQuery):	XQ1 XQ3.1

Funciones de numeración automática

generate-auto-number [altova:]

altova:generate-auto-number(ID *como xs:string*, **EmpiezaPor** *como xs:double*, **Incremento** *como xs:double*, **RestaurarAlCambiar** *como xs:string*) como `xs:integer` XP1 XP2 XQ1 XP3.1 XQ3.1

Genera un número cada vez que se llama a la función. El primer número, que se genera cuando se llama a la función por primera vez, viene dado por el argumento `EmpiezaPor`. Cada llamada posterior genera un número nuevo, que se incrementa en función del valor especificado en el argumento `Incremento`. De hecho, la función `generate-auto-number` crea un contador llamado como indique el argumento `ID` y este contador se incrementa cada vez que se llama a la función. Si el valor del argumento `RestaurarAlCambiar` cambia con respecto al valor que tenía en la llamada anterior, entonces el valor del número que se debe generar se restablece con el valor de `EmpiezaPor`. También puede restablecer la numeración automática con la función `altova:reset-auto-number`.

Ejemplo

- `altova:generate-auto-number("ChapterNumber", 1, 1, "SomeString")`

Devuelve un número cada vez que se llama a la función, empezando por `1` y con un incremento de `1` con cada llamada a función. Si el cuarto argumento continúa siendo `"SomeString"` en las llamadas posteriores, el incremento continuará. Cuando cambie el valor del cuarto argumento, se restaura el valor 1 del contador (llamado `ChapterNumber`). El valor de `ChapterNumber` también se puede restaurar llamando a la función `altova:reset-auto-number("ChapterNumber")`.

reset-auto-number [altova:]

altova:reset-auto-number(ID *como xs:string*) XP1 XP2 XQ1 XP3.1 XQ3.1

Esta función restaura el número del contador de numeración automática especificado en el argumento `ID`. El número se reemplaza con el número indicado en el argumento `EmpiezaPor` de la función `altova:generate-auto-number` que creó el contador especificado en el argumento `ID`.

Ejemplos

- `altova:reset-auto-number("ChapterNumber")` restablece el número del contador de

numeración automática llamado `ChapterNumber` que se creó con la función `altova:generate-auto-number`. El número se reemplaza con el valor del argumento `EmpiezaPor` de la función `altova:generate-auto-number` que creó `ChapterNumber`.

[Subir 454]

Funciones numéricas

hex-string-to-integer [altova:]

altova:hex-string-to-integer(CadenaHex *as xs:string*) como **xs:integer** XP3.1 XQ3.1

Toma un argumento de cadena que es el equivalente Base-16 de un entero del sistema decimal (Base-10) y devuelve un entero decimal.

Ejemplos

- **altova:hex-string-to-integer**('1') devuelve 1
- **altova:hex-string-to-integer**('9') devuelve 9
- **altova:hex-string-to-integer**('A') devuelve 10
- **altova:hex-string-to-integer**('B') devuelve 11
- **altova:hex-string-to-integer**('F') devuelve 15
- **altova:hex-string-to-integer**('G') devuelve un error
- **altova:hex-string-to-integer**('10') devuelve 16
- **altova:hex-string-to-integer**('01') devuelve 1
- **altova:hex-string-to-integer**('20') devuelve 32
- **altova:hex-string-to-integer**('21') devuelve 33
- **altova:hex-string-to-integer**('5A') devuelve 90
- **altova:hex-string-to-integer**('USA') devuelve un error

integer-to-hex-string [altova:]

altova:integer-to-hex-string(Entero *as xs:integer*) como **xs:string** XP3.1 XQ3.1

Toma el argumento **Entero** y devuelve su equivalente Base-16 en forma de cadena.

Ejemplos

- **altova:integer-to-hex-string**(1) devuelve '1'
- **altova:integer-to-hex-string**(9) devuelve '9'
- **altova:integer-to-hex-string**(10) devuelve 'A'
- **altova:integer-to-hex-string**(11) devuelve 'B'
- **altova:integer-to-hex-string**(15) devuelve 'F'
- **altova:integer-to-hex-string**(16) devuelve '10'
- **altova:integer-to-hex-string**(32) devuelve '20'
- **altova:integer-to-hex-string**(33) devuelve '21'
- **altova:integer-to-hex-string**(90) devuelve '5A'

[Subir 454]

 © 2017-2023 Altova GmbH

Funciones de formato numérico

[Subir 454]

10.2.1.6 Funciones XPath/XQuery: de esquema

Las funciones de extensión de Altova que enumeramos a continuación devuelven información del esquema. Más adelante verá descripciones de las funciones, junto con (i) ejemplos y (ii) una lista de los componentes del esquema y sus correspondientes propiedades. Estas funciones se pueden usar con los motores de Altova **XPath 3.0** y **XQuery 3.0**, y están disponibles en contextos XPath/XQuery.

Información sobre el esquema proveniente de documentos de esquema
La función **altova:schema** tiene dos argumentos: uno que no tiene argumentos y otro que tiene dos. La función que no tiene argumentos devuelve todo el esquema. A partir de ahí puede navegar por el esquema para encontrar los componentes que necesite. La función con dos argumentos devuelve un tipo concreto de componente al que se identifica por su QName. En ambos casos el valor de retorno es una función. Para ir al componente devuelto debe seleccionar una de sus propiedades. Si esta propiedad es un elemento no atómico (es decir, si es un componente), entonces puede seleccionar también una propiedad de este componente para seguir navegando. Si la propiedad seleccionada sí es un elemento atómico, entonces se devuelve el valor del elemento y no puede seguir navegando.

Nota: en las expresiones XPath de debe importar primero el esquema en el entorno de procesamiento (por ejemplo, XSLT), con la instrucción **xslt:import-schema**. En las expresiones XQuery, el esquema se debe importar de forma explícita.

Información sobre el esquema proveniente de nodos XML
La función **altova:type** envía el nodo de un documento XML y devuelve la información del tipo del modo desde el PSVI (Conjunto de información posterior a la validación de esquemas).

Nota sobre el nombre de las funciones y lenguajes

Puede utilizar todas las funciones de extensión de Altova en sus expresiones XPath/XQuery. Con ellas conseguirá funciones adicionales no disponibles en la biblioteca de funciones estándar de XPath, XQuery y XSLT. Las funciones de extensión de Altova están en el **espacio de nombres** **http://www.altova.com/xslt-extensions** y en esta sección se presentan con el prefijo, que se supone estará enlazado al espacio de nombres señalado. Tenga en cuenta que en futuras versiones del producto algunas funciones pueden dejar de ser compatibles o su comportamiento puede cambiar. Por tanto, consulte siempre la documentación del producto para conocer el funcionamiento de estas funciones en cada versión del producto.

Funciones XPath (en expresiones XPath en XSLT):	XP1 XP2 XP3.1.1
Funciones XSLT (en expresiones XPath en XSLT):	XSLT1 XSLT2 XSLT3
Funciones XQuery (en expresiones XQuery en XQuery):	XQ1 XQ3.1

altova:schema() como (function(xs:string) como item()*)? XP3.1 XQ3.1
Devuelve el componente **schema** al completo. Para navegar por este componente seleccione una de sus propiedades.

- Si esta propiedad es un componente seleccione una de sus propiedades para navegar hasta el siguiente nivel de profundidad. Puede repetir este paso para seguir navegando por el esquema.
- Si el componente es un valor atómico se devuelve este valor y no puede seguir navegando.

Las propiedades del componente **schema** son:

```
"type definitions"
"attribute declarations"
"element declarations"
"attribute group definitions"
"model group definitions"
"notation declarations"
"identity-constraint definitions"
```

Más abajo encontrará las propiedades del resto de tipos de componente.

Nota: en las expresiones XQuery, el esquema se debe importar de forma explícita. En las expresiones XPath debe importar primero el esquema en el entorno de procesamiento, por ejemplo en XSLT con la instrucción **xslt:import**.

Ejemplos

- `import schema "" at "C:\Test\ExpReport.xsd"; for $typedef in altova:schema() ("type definitions")`
  `return $typedef ("name")` devuelve los nombres de todos los tipos simples o complejos del esquema

- `import schema "" at "C:\Test\ExpReport.xsd";`
  `altova:schema() ("type definitions")[1]("name")` devuelve el nombre del primero de los tipos simples o complejos del esquema

Componentes y sus propiedades

Assertion

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Assertion"
test	Registro de propiedades XPath	

Attribute Declaration

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Attribute Declaration"

 © 2017-2023 Altova GmbH

name	Cadena	Nombre local del atributo
target namespace	Cadena	URI del espacio de nombres del atributo
type definition	Simple Type o Complex Type	
scope	Una función con propiedades ("class":"Scope", "variety": "global" o "local", "parent": el Complex Type o Attribute Group contenedor)	
value constraint	Si está presente, una función con propiedades ("class": "Value Constraint", "variety": "fixed" o "default", "value": atomic value, "lexical form": string. Tenga en cuenta que la propiedad "value" no está disponible para los tipos namespace-sensitive	
inheritable	Booleano	

- Attribute Group Declaration

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Attribute Group Definition"
name	Cadena	Nombre local del grupo de atributos
target namespace	Cadena	URI del espacio de nombres del grupo de atributos
attribute uses	Secuencia de (Attribute Use)	
attribute wildcard	Comodín de atributo opcional	

- Attribute Use

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Attribute Use"
required	Booleano	true si el atributo es obligatorio, false si es opcional
value constraint	Véase la declaración de atributos	
inheritable	Booleano	

- Attribute Wildcard

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Wildcard"
namespace constraint	Función con propiedades ("class":	

	"Namespace Constraint", "variety": "any"|"enumeration"|"not", "namespaces": secuencia de elementos xs:anyURI, "disallowed names": lista que contiene QNames y/o las cadenas "defined" y "definedSiblings"	
process contents	Cadena ("strict"|"lax"|"skip")	

- Complex Type

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Complex Type"
name	Cadena	Nombre local del tipo (vacío si es anónimo)
target namespace	Cadena	URI del espacio de nombres del tipo (vacío si es anónimo)
base type definition	Definición del Complex Type	
final	Secuencia de cadenas ("restriction"|"extension")	
context	Secuencia vacía (not implemented)	
derivation method	Cadena ("restriction"|"extension")	
abstract	Booleano	
attribute uses	Secuencia de elementos Attribute Use	
attribute wildcard	Comodín de atributo opcional	
content type	Función con propiedades: ("class":"Content Type", "variety":string ("element-only"|"empty"|"mixed"|"simple"), particle: partícula opcional, "open content": función con propiedades ("class":"Open Content", "mode": string ("interleave"|"suffix"), "wildcard": Wildcard), "simple type definition": Simple Type)	
prohibited substitutions	Secuencia de cadenas ("restriction"|"extension")	
assertions	Secuencia de elementos Assertion	

- Element Declaration

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Complex Type"
name	Cadena	Nombre local del tipo (vacío si es

© 2017-2023 Altova GmbH

		anónimo)
target namespace	Cadena	Namespace URI del tipo (vacío si es anónimo)
type definition	Simple Type o Complex Type	
type table	Función con propiedades ("class":"Type Table", "alternatives": secuencia de elementos Type Alternative, "default type definition": Simple Type o Complex Type)	
scope	Función con propiedades ("class":"Scope", "variety": ("global"|"local"), "parent": Complex Type opcional)	
value constraint	véase Attribute Declaration	
nillable	Booleano	
identity-constraint definitions	Secuencia de restricciones de identidad	
substitution group affiliations	Secuencia de declaraciones de elementos	
substitution group exclusions	Secuencia de cadenas ("restriction"|"extension")	
disallowed substitutions	Secuencia de cadenas ("restriction"|"extension"|"substitution")	
abstract	Booleano	

Element Wildcard

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Wildcard"
namespace constraint	Función con propiedades ("class": "Namespace Constraint", "variety": "any"|"enumeration"|"not", "namespaces": secuencia de xs:anyURI, "disallowed names": lista que contiene QNames y/o las cadenas "defined" y "definedSiblings"	
process contents	Cadena ("strict"|"lax"|"skip")	

Facet

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	El nombre de la faceta, por ejemplo "minLength" o "enumeration"
value	Depende de la faceta	El valor de la faceta

fixed	Booleano	
typed-value	Solo para facetas de enumeración, Array(xs:anyAtomicType*)	Una matriz que contiene los valores de la enumeración, cada uno de los cuales puede ser una secuencia de valores atómicos. (Nota: para la faceta de enumeración, la propiedad "value" es un secuencia de cadenas, independientemente del tipo)

Identity Constraint

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Identity-Constraint Definition"
name	Cadena	Nombre local de la restricción
target namespace	Cadena	URI del espacio de nombres de la restricción
identity-constraint category	Cadena ("key"|"unique"|"keyRef")	
selector	Registro de propiedades XPath	
fields	Secuencia de registros de propiedades XPath	
referenced key	(Solo para keyRef): Identity Constraint	La restricción clave correspondiente

Model Group

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Model Group"
compositor	Cadena ("sequence"|"choice"|"all")	
particles	Secuencia de partículas	

Model Group Definition

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Model Group Definition"
name	Cadena	Nombre local del grupo de modelos
target namespace	Cadena	URI del espacio de nombres del grupo de modelos
model group	Model Group	

Notation

© 2017-2023 Altova GmbH

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Notation Declaration"
name	Cadena	Nombre local de la notación
target namespace	Cadena	URI del espacio de nombres de la notación
system identifier	anyURI	
public identifier	Cadena	

Particle

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Particle"
min occurs	Número entero	
max occurs	Número entero o cadena ("unbounded")	
term	Element Declaration, Element Wildcard o ModelGroup	

Simple Type

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Simple Type Definition"
name	Cadena	Nombre local del tipo (vacío si es anónimo)
target namespace	Cadena	URI del espacio de nombres del tipo (vacío si es anónimo)
final	Secuencia de cadenas ("restriction"|"extension"|"list"|"union")	
context	Componente contenedor	
base type definition	Simple Type	
facets	Secuencia de facetas	
fundamental facets	Secuencia vacía (no implementada)	
variety	Cadena ("atomic"|"list"|"union")	
primitive type definition	Simple Type	
item type definition	(Solo para tipos de lista) Simple Type	
member type definitions	(Solo para tipos de unión) Secuencia de elementos Simple Type	

Type Alternative

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Type Alternative"
test	Registro de propiedades XPath	
type definition	Simple Type o Complex Type	

XPath Property Record

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
namespace bindings	Secuencia de funciones con propiedades ("prefix": string, "namespace": anyURI)	
default namespace	anyURI	
base URI	anyURI	El URI de base estático de la expresión XPath
expression	Cadena	La expresión XPath como cadena de texto

altova:schema(ComponentKind *as xs:string*, **Name** *as xs:QName*) **como (function(xs:string) como item()*)?** **XP3.1 XQ3.1**

Devuelve el tipo de componente que se indica en el primer argumento que tiene el mismo nombre que el que se indica en el segundo argumento. Para seguir navegando seleccione una de las propiedades del componente.

- Si esta propiedad es un componente seleccione una de sus propiedades para navegar hasta el siguiente nivel de profundidad. Puede repetir este paso para seguir navegando por el esquema.
- Si el componente es un valor atómico se devuelve este valor y no puede seguir navegando.

Nota: en las expresiones XQuery, el esquema se debe importar de forma explícita. En las expresiones XPath debe importar primero el esquema en el entorno de procesamiento, por ejemplo en XSLT con la instrucción **xslt:import**.

Ejemplos

- ```
  import schema "" at "C:\Test\ExpReport.xsd";
  altova:schema("element declaration", xs:QName("OrgChart"))("type definition")
  ("content type")("particles")[3]!.("term")("kind")
  ```
  devuelve la propiedad `kind` del término del tercer componente `particles`. Este componente desciende de la declaración de elementos que tiene un `QName` de `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")]
  ```
  devuelve, por cada **facet** de cada componente **emailType**, una matriz que contiene el tipo y el valor de ese elemento **facet**

 © 2017-2023 Altova GmbH

Componentes y sus propiedades

- Assertion

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Assertion"
test	Registro de propiedades XPath	

- Attribute Declaration

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Attribute Declaration"
name	Cadena	Nombre local del atributo
target namespace	Cadena	URI del espacio de nombres del atributo
type definition	Simple Type o Complex Type	
scope	Una función con propiedades ("class":"Scope", "variety": "global" o "local", "parent": el Complex Type o Attribute Group contenedor)	
value constraint	Si está presente, una función con propiedades ("class": "Value Constraint", "variety": "fixed" o "default", "value": atomic value, "lexical form": string. Tenga en cuenta que la propiedad "value" no está disponible para los tipos namespace-sensitive	
inheritable	Booleano	

- Attribute Group Declaration

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Attribute Group Definition"
name	Cadena	Nombre local del grupo de atributos
target namespace	Cadena	URI del espacio de nombres del grupo de atributos
attribute uses	Secuencia de (Attribute Use)	
attribute wildcard	Comodín de atributo opcional	

- Attribute Use

Nombre de la	Tipo de la propiedad	Valor de la propiedad

propiedad		
kind	Cadena	"Attribute Use"
required	Booleano	true si el atributo es obligatorio, false si es opcional
value constraint	Véase la declaración de atributos	
inheritable	Booleano	

Attribute Wildcard

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Wildcard"
namespace constraint	Función con propiedades ("class": "Namespace Constraint", "variety": "any"|"enumeration"|"not", "namespaces": secuencia de elementos xs:anyURI, "disallowed names": lista que contiene QNames y/o las cadenas "defined" y "definedSiblings"	
process contents	Cadena ("strict"|"lax"|"skip")	

Complex Type

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Complex Type"
name	Cadena	Nombre local del tipo (vacío si es anónimo)
target namespace	Cadena	URI del espacio de nombres del tipo (vacío si es anónimo)
base type definition	Definición del Complex Type	
final	Secuencia de cadenas ("restriction"|"extension")	
context	Secuencia vacía (not implemented)	
derivation method	Cadena ("restriction"|"extension")	
abstract	Booleano	
attribute uses	Secuencia de elementos Attribute Use	
attribute wildcard	Comodín de atributo opcional	
content type	Función con propiedades: ("class":"Content Type", "variety":string ("element-only"|"empty"|"mixed"|"simple"), particle: partícula opcional, "open content": función con propiedades	

© 2017-2023 Altova GmbH

	("class":"Open Content", "mode": string ("interleave"|"suffix"), "wildcard": Wildcard), "simple type definition": Simple Type)	
prohibited substitutions	Secuencia de cadenas ("restriction"|"extension")	
assertions	Secuencia de elementos Assertion	

Element Declaration

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Complex Type"
name	Cadena	Nombre local del tipo (vacío si es anónimo)
target namespace	Cadena	Namespace URI del tipo (vacío si es anónimo)
type definition	Simple Type o Complex Type	
type table	Función con propiedades ("class":"Type Table", "alternatives": secuencia de elementos Type Alternative, "default type definition": Simple Type o Complex Type)	
scope	Función con propiedades ("class":"Scope", "variety": ("global"|"local"), "parent": Complex Type opcional)	
value constraint	véase Attribute Declaration	
nillable	Booleano	
identity-constraint definitions	Secuencia de restricciones de identidad	
substitution group affiliations	Secuencia de declaraciones de elementos	
substitution group exclusions	Secuencia de cadenas ("restriction"|"extension")	
disallowed substitutions	Secuencia de cadenas ("restriction"|"extension"|"substitution")	
abstract	Booleano	

Element Wildcard

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Wildcard"

namespace constraint	Función con propiedades ("class": "Namespace Constraint", "variety": "any"|"enumeration"|"not", "namespaces": secuencia de xs:anyURI, "disallowed names": lista que contiene QNames y/o las cadenas "defined" y "definedSiblings"	
process contents	Cadena ("strict"|"lax"|"skip")	

Facet

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	El nombre de la faceta, por ejemplo "minLength" o "enumeration"
value	Depende de la faceta	El valor de la faceta
fixed	Booleano	
typed-value	Solo para facetas de enumeración, Array(xs:anyAtomicType*)	Una matriz que contiene los valores de la enumeración, cada uno de los cuales puede ser una secuencia de valores atómicos. (Nota: para la faceta de enumeración, la propiedad "value" es un secuencia de cadenas, independientemente del tipo)

Identity Constraint

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Identity-Constraint Definition"
name	Cadena	Nombre local de la restricción
target namespace	Cadena	URI del espacio de nombres de la restricción
identity-constraint category	Cadena ("key"|"unique"|"keyRef")	
selector	Registro de propiedades XPath	
fields	Secuencia de registros de propiedades XPath	
referenced key	(Solo para keyRef): Identity Constraint	La restricción clave correspondiente

Model Group

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Model Group"

 © 2017-2023 Altova GmbH

compositor	Cadena ("sequence"|"choice"|"all")	
particles	Secuencia de partículas	

Model Group Definition

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Model Group Definition"
name	Cadena	Nombre local del grupo de modelos
target namespace	Cadena	URI del espacio de nombres del grupo de modelos
model group	Model Group	

Notation

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Notation Declaration"
name	Cadena	Nombre local de la notación
target namespace	Cadena	URI del espacio de nombres de la notación
system identifier	anyURI	
public identifier	Cadena	

Particle

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Particle"
min occurs	Número entero	
max occurs	Número entero o cadena ("unbounded")	
term	Element Declaration, Element Wildcard o ModelGroup	

Simple Type

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Simple Type Definition"
name	Cadena	Nombre local del tipo (vacío si es anónimo)
target namespace	Cadena	URI del espacio de nombres del tipo (vacío si es anónimo)

final	Secuencia de cadenas ("restriction"|"extension"|"list"|"union")	
context	Componente contenedor	
base type definition	Simple Type	
facets	Secuencia de facetas	
fundamental facets	Secuencia vacía (no implementada)	
variety	Cadena ("atomic"|"list"|"union")	
primitive type definition	Simple Type	
item type definition	(Solo para tipos de lista) Simple Type	
member type definitions	(Solo para tipos de unión) Secuencia de elementos Simple Type	

Type Alternative

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Type Alternative"
test	Registro de propiedades XPath	
type definition	Simple Type o Complex Type	

XPath Property Record

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
namespace bindings	Secuencia de funciones con propiedades ("prefix": string, "namespace": anyURI)	
default namespace	anyURI	
base URI	anyURI	El URI de base estático de la expresión XPath
expression	Cadena	La expresión XPath como cadena de texto

altova:type(Node *as item?***) como (function(xs:string) como item()*)?** XP3.1 XQ3.1

La función **altova:type** indica un nodo de elemento o atributo de un documento XML y devuelve la información del tipo de nodo del PSVI (Conjunto de información posterior a la validación de esquemas).

Nota: el documento XML debe tener una declaración de esquema para que se pueda hacer referencia al esquema.

Ejemplos

- `for $element in //Email`

© 2017-2023 Altova GmbH

```
let $type := altova:type($element)
return $type
```
devuelve una función que contiene información sobre el tipo de nodo

- ```
  for $element in //Email
  let $type := altova:type($element)
  return $type ("kind")
  ```
  toma el componente de tipo del nodo (tipo simple o complejo) y devuelve el valor de la propiedad **kind** del componente

El parámetro **"_props"** devuelve las propiedades del componente seleccionado. Por ejemplo:

- ```
  for $element in //Email
  let $type := altova:type($element)
  return ($type ("kind"), $type ("_props"))
  ```
  toma el componente de tipo del nodo (tipo simple o complejo) y devuelve (i) el valor de la propiedad **kind** del componente y después (ii) las propiedades de ese componente

Componentes y sus propiedades

Assertion

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Assertion"
test	Registro de propiedades XPath	

Attribute Declaration

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Attribute Declaration"
name	Cadena	Nombre local del atributo
target namespace	Cadena	URI del espacio de nombres del atributo
type definition	Simple Type o Complex Type	
scope	Una función con propiedades ("class":"Scope", "variety": "global" o "local", "parent": el Complex Type o Attribute Group contenedor)	
value constraint	Si está presente, una función con propiedades ("class": "Value Constraint", "variety": "fixed" o "default", "value": atomic value, "lexical form": string. Tenga en cuenta que la propiedad "value" no está disponible para los tipos namespace-sensitive	

inheritable	Booleano	

Attribute Group Declaration

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Attribute Group Definition"
name	Cadena	Nombre local del grupo de atributos
target namespace	Cadena	URI del espacio de nombres del grupo de atributos
attribute uses	Secuencia de (Attribute Use)	
attribute wildcard	Comodín de atributo opcional	

Attribute Use

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Attribute Use"
required	Booleano	true si el atributo es obligatorio, false si es opcional
value constraint	Véase la declaración de atributos	
inheritable	Booleano	

Attribute Wildcard

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Wildcard"
namespace constraint	Función con propiedades ("class": "Namespace Constraint", "variety": "any"|"enumeration"|"not", "namespaces": secuencia de elementos xs:anyURI, "disallowed names": lista que contiene QNames y/o las cadenas "defined" y "definedSiblings"	
process contents	Cadena ("strict"|"lax"|"skip")	

Complex Type

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Complex Type"
name	Cadena	Nombre local del tipo (vacío si es anónimo)

© 2017-2023 Altova GmbH

target namespace	Cadena	URI del espacio de nombres del tipo (vacío si es anónimo)
base type definition	Definición del Complex Type	
final	Secuencia de cadenas ("restriction"|"extension")	
context	Secuencia vacía (not implemented)	
derivation method	Cadena ("restriction"|"extension")	
abstract	Booleano	
attribute uses	Secuencia de elementos Attribute Use	
attribute wildcard	Comodín de atributo opcional	
content type	Función con propiedades: ("class":"Content Type", "variety":string ("element-only"|"empty"|"mixed"|"simple"), particle: partícula opcional, "open content": función con propiedades ("class":"Open Content", "mode": string ("interleave"|"suffix"), "wildcard": Wildcard), "simple type definition": Simple Type)	
prohibited substitutions	Secuencia de cadenas ("restriction"|"extension")	
assertions	Secuencia de elementos Assertion	

- Element Declaration

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Complex Type"
name	Cadena	Nombre local del tipo (vacío si es anónimo)
target namespace	Cadena	Namespace URI del tipo (vacío si es anónimo)
type definition	Simple Type o Complex Type	
type table	Función con propiedades ("class":"Type Table", "alternatives": secuencia de elementos Type Alternative, "default type definition": Simple Type o Complex Type)	
scope	Función con propiedades ("class":"Scope", "variety": ("global"|"local"), "parent": Complex Type opcional)	
value constraint	véase Attribute Declaration	
nillable	Booleano	

identity-constraint definitions	Secuencia de restricciones de identidad	
substitution group affiliations	Secuencia de declaraciones de elementos	
substitution group exclusions	Secuencia de cadenas ("restriction"|"extension")	
disallowed substitutions	Secuencia de cadenas ("restriction"|"extension"|"substitution")	
abstract	Booleano	

Element Wildcard

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Wildcard"
namespace constraint	Función con propiedades ("class": "Namespace Constraint", "variety": "any"|"enumeration"|"not", "namespaces": secuencia de xs:anyURI, "disallowed names": lista que contiene QNames y/o las cadenas "defined" y "definedSiblings"	
process contents	Cadena ("strict"|"lax"|"skip")	

Facet

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	El nombre de la faceta, por ejemplo "minLength" o "enumeration"
value	Depende de la faceta	El valor de la faceta
fixed	Booleano	
typed-value	Solo para facetas de enumeración, Array(xs:anyAtomicType*)	Una matriz que contiene los valores de la enumeración, cada uno de los cuales puede ser una secuencia de valores atómicos. (Nota: para la faceta de enumeración, la propiedad "value" es un secuencia de cadenas, independientemente del tipo)

Identity Constraint

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Identity-Constraint Definition"

© 2017-2023 Altova GmbH

name	Cadena	Nombre local de la restricción
target namespace	Cadena	URI del espacio de nombres de la restricción
identity-constraint category	Cadena ("key"|"unique"|"keyRef")	
selector	Registro de propiedades XPath	
fields	Secuencia de registros de propiedades XPath	
referenced key	(Solo para keyRef): Identity Constraint	La restricción clave correspondiente

Model Group

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Model Group"
compositor	Cadena ("sequence"|"choice"|"all")	
particles	Secuencia de partículas	

Model Group Definition

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Model Group Definition"
name	Cadena	Nombre local del grupo de modelos
target namespace	Cadena	URI del espacio de nombres del grupo de modelos
model group	Model Group	

Notation

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Notation Declaration"
name	Cadena	Nombre local de la notación
target namespace	Cadena	URI del espacio de nombres de la notación
system identifier	anyURI	
public identifier	Cadena	

Particle

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad

kind	Cadena	"Particle"
min occurs	Número entero	
max occurs	Número entero o cadena ("unbounded")	
term	Element Declaration, Element Wildcard o ModelGroup	

Simple Type

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Simple Type Definition"
name	Cadena	Nombre local del tipo (vacío si es anónimo)
target namespace	Cadena	URI del espacio de nombres del tipo (vacío si es anónimo)
final	Secuencia de cadenas ("restriction"|"extension"|"list"|"union")	
context	Componente contenedor	
base type definition	Simple Type	
facets	Secuencia de facetas	
fundamental facets	Secuencia vacía (no implementada)	
variety	Cadena ("atomic"|"list"|"union")	
primitive type definition	Simple Type	
item type definition	(Solo para tipos de lista) Simple Type	
member type definitions	(Solo para tipos de unión) Secuencia de elementos Simple Type	

Type Alternative

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Type Alternative"
test	Registro de propiedades XPath	
type definition	Simple Type o Complex Type	

XPath Property Record

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
namespace bindings	Secuencia de funciones con propiedades ("prefix": string, "namespace": anyURI)	

© 2017-2023 Altova GmbH

default namespace	anyURI	
base URI	anyURI	El URI de base estático de la expresión XPath
expression	Cadena	La expresión XPath como cadena de texto

10.2.1.7 Funciones XPath/XQuery: de secuencia

Las funciones de extensión de Altova para trabajar con secuencias pueden utilizarse en expresiones XPath y XQuery y ofrecen funciones adicionales para el procesamiento de datos. Estas funciones se pueden usar con los motores **XPath 3.0** y **XQuery 3.0** de Altova. Están disponibles en contextos XPath/XQuery.

Nota sobre el nombre de las funciones y lenguajes

Puede utilizar todas las funciones de extensión de Altova en sus expresiones XPath/XQuery. Con ellas conseguirá funciones adicionales no disponibles en la biblioteca de funciones estándar de XPath, XQuery y XSLT. Las funciones de extensión de Altova están en el **espacio de nombres** `http://www.altova.com/xslt-extensions` y en esta sección se presentan con el prefijo, que se supone estará enlazado al espacio de nombres señalado. Tenga en cuenta que en futuras versiones del producto algunas funciones pueden dejar de ser compatibles o su comportamiento puede cambiar. Por tanto, consulte siempre la documentación del producto para conocer el funcionamiento de estas funciones en cada versión del producto.

Funciones XPath (en expresiones XPath en XSLT):	XP1 XP2 XP3.1.1
Funciones XSLT (en expresiones XPath en XSLT):	XSLT1 XSLT2 XSLT3
Funciones XQuery (en expresiones XQuery en XQuery):	XQ1 XQ3.1

▼ attributes [altova:]

altova:attributes(NombreAtributo *as xs:string***)** como **attribute()*** XP3.1 XQ3.1
Devuelve todos los atributos cuyo nombre local coincida con el nombre dado como argumento de entrada (`NombreAtributo`). La búsqueda tiene en cuenta el uso de mayúsculas y minúsculas y se lleva a cabo en el eje `attribute::`.

▣ *Ejemplos*

- **altova:attributes**("MiAtributo") devuelve `MiAtributo()*`

altova:attributes(NombreAtributo *as xs:string*, **OpcionesBúsqueda** *as xs:string***)** como **attribute()*** XP3.1 XQ3.1
Devuelve todos los atributos cuyo nombre local coincida con el nombre dado como argumento de entrada

(NombreAtributo). La búsqueda tiene en cuenta el uso de mayúsculas y minúsculas y se lleva a cabo en el eje attribute::. El segundo argumento es una cadena con marcas de búsqueda. Estas son las marcas disponibles:

r = habilita la búsqueda de expresiones regulares. En este caso, NombreAtributo debe ser una cadena de búsqueda de expresión regular;
i = la búsqueda no tiene en cuenta el uso de mayúsculas y minúsculas;
p = incluye el prefijo de espacio de nombres en la búsqueda. En este caso, NombreAtributo debe contener el prefijo de espacio de nombres (p. ej.: MiAtributo).

Las marcas pueden escribirse en cualquier orden y no hace falta utilizar todas. Si usa marcas no válidas, se genera un error. También puede usar una cadena vacía para el segundo argumento. Esto tiene el mismo efecto que usar solo el primer argumento. Sin embargo, no está permitido usar una secuencia vacía.

Ejemplos

- **altova:attributes**("MiAtributo", "rip") devuelve MiAtributo()*
- **altova:attributes**("MiAtributo", "pri") devuelve MiAtributo()*
- **altova:attributes**("MiAtributo", "") devuelve MiAtributo()*
- **altova:attributes**("MiAtributo", "Rip") devuelve un error de marca desconocida.
- **altova:attributes**("MiAtributo",) devuelve un error diciendo que falta el segundo argumento.

elements [altova:]

altova:elements(NombreElemento *as xs:string***)** como **elemento()*** **XP3.1** **XQ3.1**
Devuelve todos los elementos cuyo nombre local coincida con el nombre dado como argumento de entrada (NombreElemento). La búsqueda tiene en cuenta el uso de mayúsculas y minúsculas y se lleva a cabo en el eje child::.

Ejemplos

- **altova:elements**("MiElemento") devuelve MiElemento()*

altova:elements(NombreElemento *as xs:string*, **OpcionesBúsqueda** *as xs:string***)** como **elemento()*** **XP3.1** **XQ3.1**
Devuelve todos los elementos cuyo nombre local coincida con el nombre dado como argumento de entrada (NombreElemento). La búsqueda tiene en cuenta el uso de mayúsculas y minúsculas y se lleva a cabo en el eje child::. El segundo argumento es una cadena con marcas de búsqueda. Estas son las marcas disponibles:

r = habilita la búsqueda de expresiones regulares. En este caso, NombreElemento debe ser una cadena de búsqueda de expresión regular;
i = la búsqueda no tiene en cuenta el uso de mayúsculas y minúsculas;
p = incluye el prefijo de espacio de nombres en la búsqueda. En este caso, NombreElemento debe contener el prefijo de espacio de nombres (p. ej.: MiElemento).

Las marcas pueden escribirse en cualquier orden y no hace falta utilizar todas. Si usa marcas no válidas, se genera un error. También puede usar una cadena vacía para el segundo argumento. Esto tiene el mismo efecto que usar solo el primer argumento. Sin embargo, no está permitido usar una secuencia

 © 2017-2023 Altova GmbH

vacía.

Ejemplos

- **altova:elements**("MiElemento", "rip") devuelve MiElemento()*
- **altova:elements**("MiElemento", "pri") devuelve MiElemento()*
- **altova:elements**("MiElemento", "") devuelve MiElemento()*
- **altova:elements**("MiElemento", "Rip") devuelve un error de marca desconocida.
- **altova:elements**("MiElemento",) devuelve un error diciendo que falta el segundo argumento.

find-first [altova:]

altova:find-first((Secuencia *()**)*, **(Condición(Elemento-Secuencia** *como xs:boolean*))
como **item()?** **XP3.1** **XQ3.1**
Esta función toma dos argumentos. El primero es una secuencia de uno o varios elementos de cualquier tipo de datos. El segundo argumento, **Condición**, es una referencia a una función XPath que toma un argumento (es decir, su aridad es 1) y devuelve un valor binario. Cada elemento de **Secuencia** se envía a su vez a la función a la que se hace referencia en **Condición**. Nota: recuerde que esta función solo toma un argumento. El primer elemento de **Secuencia** que consiga que la función de **Condición** dé **true()** como resultado se devuelve como resultado de **find-first** y la iteración se detiene.

Ejemplos

- **altova:find-first**(5 to 10, function($a) {$a mod 2 = 0}) devuelve xs:integer 6
  El argumento **Condición** remite a la función inline XPath 3.0 **function()**, que declara una función inline llamada **$a** y después la define. Cada elemento del argumento Secuencia de **find-first** se envía a su vez como valor de entrada a **$a**. El valor de entrada se prueba en la condición en la definición de función ($a mod 2 = 0). El primer valor de entrada que cumpla la condición se devuelve como resultado de **find-first** (en este caso **6**).

- **altova:find-first**((1 to 10), (function($a) {$a+3=7})) devuelve xs:integer 4

Más ejemplos
Si existe el archivo **C:\Temp\Customers.xml**:

- **altova:find-first**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) devuelve xs:string C:\Temp\Customers.xml

Si no existe el archivo **C:\Temp\Customers.xml** pero existe **http://www.altova.com/index.html**:

- **altova:find-first**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) devuelve xs:string http://www.altova.com/index.html

Si no existe el archivo **C:\Temp\Customers.xml** y tampoco existe **http://www.altova.com/index.html**:

- **altova:find-first**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) no devuelve ningún resultado

Notas sobre los ejemplos anteriores

- La función XPath 3.0 `doc-available` toma un solo argumento de cadena, que se usa como URI, y devuelve `true` si en el URI dado se encuentra un nodo de documento. El documento que está en el URI dado debe ser un documento XML.
- La función `doc-available` se puede usar para **Condición**, el segundo argumento de `find-first`, porque solamente toma un argumento (aridad=1), porque toma un `item()` como entrada (una cadena que se usa como URI) y devuelve un valor binario.
- Recuerde que solamente se hace referencia a la función `doc-available` pero no se le llama. El sufijo `#1` que se anexa a la función indica una función cuya aridad es 1. Es decir, `doc-available#1` simplemente significa "*Utilizar la función doc-availabe() que tiene aridad=1, pasándole como solo argumento a su vez cada uno de los elementos de la primera secuencia.*" Como resultado, se pasarán las dos cadenas a **doc-available()**, que utiliza la cadena como URI y prueba si existe un nodo de documento en el URI. Si existe, entonces `doc-available()` da como resultado `true()` y esa cadena se devuelve como resultado de la función **find-first**. Nota sobre la función `doc-available()`: las rutas de acceso relativas se resuelven en relación al URI base actual, que es por defecto el URI del documento XML desde el que se carga la función.

▼ find-first-combination [altova:]

altova:find-first-combination((Sec-01 *como item()**)**, (Sec-02** *como item()**)**, (Condición(Elem-Sec-01, Elem-Sec-02** *como xs:boolean*)) como **item()*** XP3.1 XQ3.1

Esta función toma tres argumentos:

- Los dos primeros (**Sec-01** y **Sec-02**) son secuencias de uno o más elementos de cualquier tipo de datos.
- El tercero (**Condición**) es una referencia a una función XPath que toma dos argumentos (su aridad es 2) y devuelve un valor binario.

Los elementos de **Sec-01** y **Sec-02** se pasan en pares ordenados (cada par está formado por un elemento de cada secuencia) como argumentos de la función de **Condición**. Los pares se ordenan de la siguiente manera:

```
Si   Sec-01 = X1, X2, X3 ... Xn
Y  Sec-02 = Y1, Y2, Y3 ... Yn
Entonces (X1 Y1), (X1 Y2), (X1 Y3) ... (X1 Yn), (X2 Y1), (X2 Y2) ... (Xn Yn)
```

El primer par ordenado que consiga que la función de **Condición** dé como resultado **true()** se devuelve como resultado de **find-first-combination**. Recuerde que (i) si la función de **Condición** recorre los pares de argumentos dados y no consigue dar **true()** como resultado ni una vez, entonces **find-first-combination** devuelve *Sin resultados;* (ii) el resultado de **find-first-combination** siempre será un par de elementos (de cualquier tipo de datos) o ningún elemento.

▬ *Ejemplos*

- **altova:find-first-pair**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 32}) devuelve la secuencia de `xs:integers (11, 21)`
- **altova:find-first-pair**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 33}) devuelve la secuencia de `xs:integers (11, 22)`
- **altova:find-first-pair**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 34}) devuelve

 © 2017-2023 Altova GmbH

la secuencia de `xs:integers (11, 23)`

▼ find-first-pair [altova:]

altova:find-first-pair((Sec-01 *como item()**)**, (Sec-02** *como item()**)**, (Condición(Elem-Sec-01, Elem-Sec-02** *como xs:boolean*)) como **item()*** XP3.1 XQ3.1
Esta función toma tres argumentos:

- Los dos primeros (**Sec-01** y **Sec-02**) son secuencias de uno o más elementos de cualquier tipo de datos.
- El tercero (**Condición**) es una referencia a una función XPath que toma dos argumentos (su aridad es 2) y devuelve un valor binario.

Los elementos de **Sec-01** y **Sec-02** se pasan en pares ordenados como argumentos de la función de **Condición**. Los pares se ordenan de la siguiente manera:

```
Si   Sec-01 = X1, X2, X3 ... Xn
Y  Sec-02 = Y1, Y2, Y3 ... Yn
Entonces (X1 Y1), (X2 Y2), (X3 Y3) ... (Xn Yn)
```

El primer par ordenado que consiga que la función de **Condición** dé como resultado **true()** se devuelve como resultado de **find-first-pair**. Recuerde que (i) si la función de **Condición** recorre los pares de argumentos dados y no consigue dar **true()** como resultado ni una vez, entonces **find-first-pair** devuelve *Sin resultados;* (ii) el resultado de **find-first-pair** siempre será un par de elementos (de cualquier tipo de datos) o ningún elemento.

▣ *Ejemplos*

- **altova:find-first-pair**`(11 to 20, 21 to 30, function($a, $b) {$a+$b = 32})` devuelve la secuencia de `xs:integers (11, 21)`
- **altova:find-first-pair**`(11 to 20, 21 to 30, function($a, $b) {$a+$b = 33})` devuelve *Sin resultados*

Observe que en los dos ejemplos anteriores el orden de los pares es: `(11, 21) (12, 22) (13, 23)...(20, 30)`. Por ese motivo el segundo ejemplo no obtiene resultados (porque ningún par ordenado consigue sumar `33`).

▼ find-first-pair-pos [altova:]

altova:find-first-pair-pos((Sec-01 *como item()**)**, (Sec-02** *como item()**)**, (Condición(Elem-Sec-01, Elem-Sec-02** *como xs:boolean*)) como **xs:integer** XP3.1 XQ3.1
Esta función toma tres argumentos:

- Los dos primeros (**Sec-01** y **Sec-02**) son secuencias de uno o más elementos de cualquier tipo de datos.
- El tercero (**Condición**) es una referencia a una función XPath que toma dos argumentos (su aridad es 2) y devuelve un valor binario.

Los elementos de **Sec-01** y **Sec-02** se pasan en pares ordenados como argumentos de la función de **Condición**. Los pares se ordenan de la siguiente manera:

```
Si   Sec-01 = X1, X2, X3 ... Xn
Y  Sec-02 = Y1, Y2, Y3 ... Yn
Entonces (X1 Y1), (X2 Y2), (X3 Y3) ... (Xn Yn)
```

La posición de índice del primer par ordenado que consiga que la función de **Condición** dé como resultado **true()** se devuelve como resultado de **find-first-pair-pos**. Recuerde que si la función de **Condición** recorre los pares de argumentos dados y no da como resultado **true()** ni una sola vez, entonces **find-first-pair-pos** devuelve *Sin resultados*.

Ejemplos

- **altova:find-first-pair**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 32}) devuelve 1
- **altova:find-first-pair**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 33}) devuelve *Sin resultados*

Observe que en los dos ejemplos anteriores el orden de los pares es: (11, 21) (12, 22) (13, 23)...(20, 30). En el primer ejemplo el primer par consigue que la función de **Condición** dé como resultado **true()** y, por tanto, se devuelve la posición de índice que tienen en la secuencia (**1**). El segundo ejemplo, sin embargo, devuelve *Sin resultados* porque ningún par consigue sumar 33.

find-first-pos [altova:]

altova:find-first-pos((Secuencia *como item()**)**,* **(Condición(Elem-Sec** *como xs:boolean***))** como **xs:integer** **XP3.1** **XQ3.1**

Esta función toma dos argumentos. El primer argumento es una secuencia de uno o varios elementos de cualquier tipo. El segundo argumento (**Condición**) es una referencia a una función XPath que toma un argumento (su aridad es 1) y devuelve un valor binario. Cada elemento de **Secuencia** se envía a su vez a la función a la que se hace referencia en **Condición**. (Recuerde que esta función toma un solo argumento.) El primer elemento de **Secuencia** que consiga que la función de **Condición** dé como resultado **true()** devuelve la posición de índice que tiene en **Secuencia** como resultado de **find-first-pos** y la iteración se detiene.

Ejemplos

- **altova:find-first-pos**(5 to 10, function($a) {$a mod 2 = 0}) devuelve xs:integer 2

El argumento **Condición** hace referencia a la función inline XPath 3.0 **function()**, que declara una función inline llamada **$a** y después la define. Cada elemento del argumento **Sequence** de **find-first-pos** se pasa a su vez como valor de entrada de **$a**. El valor de entrada se prueba en la condición de la definición de función ($a mod 2 = 0). La posición de índice que tiene en la secuencia el primer valor de entrada que cumple la condición se devuelve como resultado de **find-first-pos** (en este caso es la posición de índice **2**, porque **6** es el primer valor (de la secuencia) que cumple la condición y su posición de índice en la secuencia es **2**).

- **altova:find-first-pos**((2 to 10), (function($a) {$a+3=7})) devuelve xs:integer 3

Más ejemplos

Si existe el archivo **C:\Temp\Customers.xml**:

© 2017-2023 Altova GmbH

- **altova:find-first-pos**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) devuelve 1

Si no existe el archivo **C:\Temp\Customers.xml** pero existe **http://www.altova.com/index.html**:

- **altova:find-first-pos**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) devuelve 2

Si no existe el archivo **C:\Temp\Customers.xml** y tampoco existe **http://www.altova.com/index.html**:

- **altova:find-first-pos**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) no devuelve ningún resultado

Notas sobre los ejemplos anteriores

- La función XPath 3.0 doc-available toma un solo argumento de cadena, que se usa como URI, y devuelve true si en el URI dado se encuentra un nodo de documento. El documento que está en el URI dado debe ser un documento XML.
- La función doc-available se puede usar para **Condición**, el segundo argumento de find-first-pos, porque solamente toma un argumento (aridad=1), porque toma un item() como entrada (una cadena que se usa como URI) y devuelve un valor binario.
- Recuerde que solamente se hace referencia a la función doc-available pero no se le llama. El sufijo #1 que se anexa a la función indica una función cuya aridad es 1. Es decir, doc-available#1 simplemente significa "*Utilizar la función doc-availabe() que tiene aridad=1, pasándole como solo argumento a su vez cada uno de los elementos de la primera secuencia.*" Como resultado, se pasarán las dos cadenas a **doc-available()**, que utiliza la cadena como URI y prueba si existe un nodo de documento en el URI. Si existe, entonces doc-available() da como resultado true() y esa cadena se devuelve como resultado de la función **find-first-pos**. Nota sobre la función doc-available(): las rutas de acceso relativas se resuelven en relación al URI base actual, que es por defecto el URI del documento XML desde el que se carga la función.

for-each-attribute-pair [altova:]

altova:for-each-attribute-pair(Seq1 *como element()?*, **Seq2** *como element()?*, **Function** *como function()*) como **item()*** **XP3.1** **XQ3.1**

Los primeros dos argumentos identifican dos elementos cuyos atributos se usan para construir pares de atributos donde uno de los atributos del par se obtiene del primer elemento y el otro atributo del segundo elemento. Los pares de atributos se seleccionan basándose en que tienen el mismo nombre y se ordenan alfabéticamente por grupos. Si un atributo no tiene un atributo correspondiente en el otro elemento, entonces el par está "desarticulado", lo que significa que tiene un solo miembro. El elemento de la función (tercer argumento Function) se aplica por separado a cada par de la secuencia de pares (articulados y desarticulados) y el resultado es una secuencia de elementos.

Ejemplos

- **altova:for-each-attribute-pair**(/Example/Test-A, /Example/Test-B, function($a, $b) {$a+b}) devuelve...

`(2, 4, 6)` si
`<Test-A att1="1" att2="2" att3="3" />`
`<Test-B att1="1" att2="2" att3="3" />`

`(2, 4, 6)` si
`<Test-A att2="2" att1="1" att3="3" />`
`<Test-B att3="3" att2="2" att1="1" />`

`(2, 6)` si
`<Test-A att4="4" att1="1" att3="3" />`
`<Test-B att3="3" att2="2" att1="1" />`

Nota: El resultado `(2, 6)` *se obtiene mediante la siguiente acción:* `(1+1, ()+2, 3+3, 4+())`. *Si uno de los operandos es la secuencia vacía, como en el caso de los elementos 2 y 4, entonces el resultado de la suma es una secuencia vacía.*

- **`altova:for-each-attribute-pair`**`(/Example/Test-A, /Example/Test-B, concat#2)` devuelve...

`(11, 22, 33)` si
`<Test-A att1="1" att2="2" att3="3" />`
`<Test-B att1="1" att2="2" att3="3" />`

`(11, 2, 33, 4)` si
`<Test-A att4="4" att1="1" att3="3" />`
`<Test-B att3="3" att2="2" att1="1" />`

for-each-combination [altova:]

`altova:for-each-combination(PrimeraSecuencia` *como item()**, **`SegundaSecuencia`** *como item()**, **`función($i,$j){$i || $j} )`** como **`item()*`** **XP3.1** **XQ3.1**

Los elementos de las dos secuencias en los primeros dos argumentos se combinan de forma que el primer elemento de la primera secuencia se combina, en orden, una vez con cada elemento de la segunda secuencia. La función dada como tercer argumento se aplica a cada una de las combinaciones de la secuencia resultante y da como resultado una secuencia de elementos (*véase ejemplo*).

Ejemplos

- **`altova:for-each-combination`**`( ('a', 'b', 'c'), ('1', '2', '3'), function($i, $j) {$i || $j} )` devuelve `('a1', 'a2', 'a3', 'b1', 'b2', 'b3', 'c1', 'c2', 'c3')`

for-each-matching-attribute-pair [altova:]

`altova:for-each-matching-attribute-pair(Seq1` *como element()?*, **`Seq2`** *como element()?*, **`Function`** *como function()`**)** como **`item()*`** **XP3.1** **XQ3.1**

Los primeros dos argumentos identifican dos elementos cuyos atributos se usan para construir pares de atributos donde un atributo de cada par se obtiene del primer elemento y el otro atributo del par se obtiene del segundo elemento. Los pares de elementos se seleccionan basándose en que tienen el mismo nombre y se ordenan alfabéticamente por grupos. Si un atributo no tiene un atributo correspondiente en el otro elemento, entonces no se construye ningún par. El elemento de la función (tercer argumento

 © 2017-2023 Altova GmbH

Function) se aplica por separado a cada par de la secuencia de pares (articulados y desarticulados) y el resultado es una secuencia de elementos.

Ejemplos

- **altova:for-each-matching-attribute-pair**(/Example/Test-A, /Example/Test-B, function($a, $b){$a+b}) devuelve...

```
(2, 4, 6) if
<Test-A att1="1" att2="2" att3="3" />
<Test-B att1="1" att2="2" att3="3" />

(2, 4, 6) if
<Test-A att2="2" att1="1" att3="3" />
<Test-B att3="3" att2="2" att1="1" />

(2, 6) if
<Test-A att4="4" att1="1" att3="3" />
<Test-B att3="3" att2="2" att3="1" />
```

- **altova:for-each-matching-attribute-pair**(/Example/Test-A, /Example/Test-B, concat#2) devuelve...

```
(11, 22, 33) if
<Test-A att1="1" att2="2" att3="3" />
<Test-B att1="1" att2="2" att3="3" />

(11, 33) if
<Test-A att4="4" att1="1" att3="3" />
<Test-B att3="3" att2="2" att1="1" />
```

▼ substitute-empty [altova:]

altova:substitute-empty(PrimeraSecuencia *as item()**, **SegundaSecuencia** *as item()***)** como **item()*** XP3.1 XQ3.1

Si **PrimeraSecuencia** está vacío, la función devuelve **SegundaSecuencia**. Si **PrimeraSecuencia** no está vacío, la función devuelve **PrimeraSecuencia**.

Ejemplos

- **altova:substitute-empty**((1,2,3), (4,5,6)) devuelve (1,2,3)
- **altova:substitute-empty**((), (4,5,6)) devuelve (4,5,6)

10.2.1.8 Funciones XPath/XQuery: de cadena

Las funciones de extensión de Altova para trabajar con cadenas pueden utilizarse en expresiones XPath y XQuery y ofrecen funciones adicionales para el procesamiento de datos. Estas funciones se pueden usar con los motores **XPath 3.0** y **XQuery 3.0** de Altova. Están disponibles en contextos XPath/XQuery.

Nota sobre el nombre de las funciones y lenguajes

Puede utilizar todas las funciones de extensión de Altova en sus expresiones XPath/XQuery. Con ellas conseguirá funciones adicionales no disponibles en la biblioteca de funciones estándar de XPath, XQuery y XSLT. Las funciones de extensión de Altova están en el **espacio de nombres** `http://www.altova.com/xslt-extensions` y en esta sección se presentan con el prefijo, que se supone estará enlazado al espacio de nombres señalado. Tenga en cuenta que en futuras versiones del producto algunas funciones pueden dejar de ser compatibles o su comportamiento puede cambiar. Por tanto, consulte siempre la documentación del producto para conocer el funcionamiento de estas funciones en cada versión del producto.

Funciones XPath (en expresiones XPath en XSLT):	XP1 XP2 XP3.1.1
Funciones XSLT (en expresiones XPath en XSLT):	XSLT1 XSLT2 XSLT3
Funciones XQuery (en expresiones XQuery en XQuery):	XQ1 XQ3.1

camel-case [altova:]

altova:camel-case(CadenaEntrada *como xs:string***)** como **xs:string** XP3.1 XQ3.1
Devuelve la cadena de entrada **CadenaEntrada** escrita en CamelCase. La cadena se analiza usando la expresión regular `'\s'` (que es la forma abreviada del carácter espacio en blanco). El primer carácter que no sea un espacio en blanco situado después de un espacio en blanco o de una secuencia de espacios en blanco consecutivos se pondrá en mayúsculas. El primer carácter de la cadena de salida se pondrá en mayúsculas.

Ejemplos

- **altova:camel-case**("max") devuelve Max
- **altova:camel-case**("max max") devuelve Max Max
- **altova:camel-case**("file01.xml") devuelve File01.xml
- **altova:camel-case**("file01.xml file02.xml") devuelve File01.xml File02.xml
- **altova:camel-case**("file01.xml file02.xml") devuelve File01.xml File02.xml
- **altova:camel-case**("file01.xml -file02.xml") devuelve File01.xml -file02.xml

altova:camel-case(CadenaEntrada *como xs:string*, **CaracteresDivisión** *como xs:string*, **EsExpReg** *como xs:boolean***)** como **xs:string** XP3.1 XQ3.1
Devuelve la cadena de entrada **CadenaEntrada** escrita en CamelCase usando los **CaracteresDivisón** para determinar qué caracteres desencadenan el siguiente uso de mayúsculas. El argumento **CaracteresDivisón** se usa como expresión regular **cuando EsExpReg = true()** o como caracteres planos cuando r **EsExpReg = false()**. El primer carácter de la cadensa de salida se escribe con mayúsculas.

Ejemplos

- **altova:camel-case**("setname getname", "set|get", true()) devuelve setName getName
- **altova:camel-case**("altova\documents\testcases", "\", false()) devuelve Altova\Documents\Testcases

 © 2017-2023 Altova GmbH

char [altova:]

altova:char(Posición *as xs:integer***)** como **xs:string** XP3.1 XQ3.1
Devuelve una cadena que contiene el carácter que está en la posición indicada por el argumento **Posición** en la cadena que se obtiene al convertir el valor del elemento de contexto en `xs:string`. La cadena resultante estará vacía si en la posición indicada no existe ningún carácter.

Ejemplos

Si el elemento de contexto es **1234ABCD**:

- **altova:char**(2) devuelve 2
- **altova:char**(5) devuelve A
- **altova:char**(9) devuelve la cadena vacía
- **altova:char**(-2) devuelve la cadena vacía

altova:char(CadenaEntrada *as xs:string*, **Posición** *as xs:integer***)** como **xs:string** XP3.1 XQ3.1
Devuelve una cadena que contiene el carácter que está en la posición indicada por el argumento **Posición** en la cadena dada por el argumento **CadenaEntrada**. La cadena resultante estará vacía si en la posición indicada no existe ningún carácter.

Ejemplos

- **altova:char**("2014-01-15", 5) devuelve -
- **altova:char**("USA", 1) devuelve U
- **altova:char**("USA", 1) devuelve la cadena vacía
- **altova:char**("USA", -2) devuelve la cadena vacía

create-hash-from-string[altova:]

altova:create-hash-from-string(InputString *como xs:string***)** como **xs:string** XP2 XQ1 XP3.1 XQ3.1
altova:create-hash-from-string(InputString *como xs:string*, **HashAlgo** *as xs:string***)** como **xs:string** XP2 XQ1 XP3.1 XQ3.1
Genera una cadena hash a partir de InputString usando el algoritmo de hash especificado por el argumento `HashAlgo`. Se pueden usar los siguientes algoritmos de hash (en mayúsculas o minúsculas): **MD5**, **SHA-1**, **SHA-224**, **SHA-256**, **SHA-384**, **SHA-512**. Si no se especifica el segundo argumento (*véase la primera instrucción*) se usa el algoritmo de hash **SHA-256**.

Ejemplos

- **altova:create-hash-from-string**('abc') devuelve una cadena hash generada usando el algoritmo de hash **SHA-256**.
- **altova:create-hash-from-string**('abc', 'md5') devuelve una cadena hash generada usando el algoritmo de hash **MD5**.
- **altova:create-hash-from-string**('abc', 'MD5') devuelve una cadena hash generada usando el algoritmo de hash **MD5**.

first-chars [altova:]

altova:first-chars(X *as xs:integer*) como **xs:string** XP3.1 XQ3.1
Devuelve una cadena que contiene los **X** primeros caracteres de la cadena que se obtiene al convertir el valor del elemento de contexto en `xs:string`.

Ejemplos

Si el elemento de contexto es **1234ABCD**:

- **altova:first-chars**(2) devuelve 12
- **altova:first-chars**(5) devuelve 1234A
- **altova:first-chars**(9) devuelve 1234ABCD

altova:first-chars(CadenaEntrada *as xs:string*, **X** *as xs:integer*) como **xs:string** XP3.1 XQ3.1
Devuelve una cadena que contiene los **X** primeros caracteres de la cadena dada como argumento **CadenaEntrada**.

Ejemplos

- **altova:first-chars**("2014-01-15", 5) devuelve 2014-
- **altova:first-chars**("USA", 1) devuelve U

format-string [altova:]

altova:format-string(InputString *como xs:string*, **FormatSequence** *como item()**) como **xs:string** XP3.1 XQ3.1
La cadena de entrada (primer argumento) contiene parámetros posicionales (%1, %2, etc). Cada parámetro es reemplazado por el elemento cadena ubicado en la posición correspondiente de la secuencia de formato (enviada como segundo argumento). Por tanto, el primer elemento de la secuencia de formato reemplaza al parámetro posicional %1, el segundo elemento reemplaza a %2 y así sucesivamente. La función devuelve esta secuencia con formato que contiene los elementos de reemplazo. Si no existe una cadena para alguno de los parámetros posicionales, entonces se devuelve ese mismo parámetro posicional. Esto ocurre cuando el índice de un parámetro posicional es mayor que el número de elementos de la secuencia de formato.

Ejemplos

- **altova:format-string**('Hello %1, %2, %3', ('Jane','John','Joe')) devuelve "Hello Jane, John, Joe"
- **altova:format-string**('Hello %1, %2, %3', ('Jane','John','Joe', 'Tom')) devuelve "Hello Jane, John, Joe"
- **altova:format-string**('Hello %1, %2, %4', ('Jane','John','Joe', 'Tom')) devuelve "Hello Jane, John, Tom"
- **altova:format-string**('Hello %1, %2, %4', ('Jane','John','Joe')) devuelve "Hello Jane, John, %4"

last-chars [altova:]

altova:last-chars(X *as xs:integer*) como **xs:string** XP3.1 XQ3.1
Devuelve una cadena que contiene los X últimos caracteres de la cadena que se obtiene al convertir el valor del elemento de contexto en `xs:string`.

© 2017-2023 Altova GmbH

Ejemplos

Si el elemento de contexto es **1234ABCD**:

- **altova:last-chars**(2) devuelve CD
- **altova:last-chars**(5) devuelve 4ABCD
- **altova:last-chars**(9) devuelve 1234ABCD

altova:last-chars(CadenaEntrada *as xs:string*, **X** *as xs:integer*) como **xs:string** XP3.1 XQ3.1
Devuelve una cadena que contiene los **X** últimos caracteres de la cadena dada como argumento **CadenaEntrada**.

Ejemplos

- **altova:last-chars**("2014-01-15", 5) devuelve 01-15-
- **altova:last-chars**("USA", 10) devuelve USA

pad-string-left [altova:]

altova:pad-string-left(CadenaParaRellenar *como xs:string*, **LongitudCadena** *como xs:integer*, **CarácterRelleno** *como xs:string*) como **xs:string** XP3.1 XQ3.1
El argumento CarácterRelleno es un solo carácter. Se añade a la izquierda de la cadena para aumentar el número de caracteres de la CadenaParaRellenar, de modo que este número equivalga al valor entero del argumento LongitudCadena. El argumento LongitudCadena puede tener cualquier valor entero (positivo o negativo), pero el relleno solo se lleva a cabo si el valor de LongitudCadena es mayor que el número de caracteres de CadenaParaRellenar. Si CadenaParaRellenar tiene más caracteres que el valor de LongitudCadena, entonces CadenaParaRellenar se deja como está.

Ejemplos

- **altova:pad-string-left**('AP', 1, 'Z') devuelve 'AP'
- **altova:pad-string-left**('AP', 2, 'Z') devuelve 'AP'
- **altova:pad-string-left**('AP', 3, 'Z') devuelve 'ZAP'
- **altova:pad-string-left**('AP', 4, 'Z') devuelve 'ZZAP'
- **altova:pad-string-left**('AP', -3, 'Z') devuelve 'AP'
- **altova:pad-string-left**('AP', 3, 'YZ') devuelve un error indicando que el carácter de relleno es demasiado largo.

pad-string-right [altova:]

altova:pad-string-right(CadenaParaRellenar *como xs:string*, **LongitudCadena** *como xs:integer*, **CarácterRelleno** *como xs:string*) como **xs:string** XP3.1 XQ3.1
El argumento CarácterRelleno es un solo carácter. Se añade a la derecha de la cadena para aumentar el número de caracteres de la CadenaParaRellenar, de modo que este número equivalga al valor entero del argumento LongitudCadena. El argumento LongitudCadena puede tener cualquier valor entero (positivo o negativo), pero el relleno solo se lleva a cabo si el valor de LongitudCadena es mayor que el número de caracteres de CadenaParaRellenar. Si CadenaParaRellenar tiene más caracteres que el valor de LongitudCadena, entonces CadenaParaRellenar se deja como está.

Ejemplos

- `altova:pad-string-right('AP', 1, 'Z')` devuelve `'AP'`
- `altova:pad-string-right('AP', 2, 'Z')` devuelve `'AP'`
- `altova:pad-string-right('AP', 3, 'Z')` devuelve `'APZ'`
- `altova:pad-string-right('AP', 4, 'Z')` devuelve `'APZZ'`
- `altova:pad-string-right('AP', -3, 'Z')` devuelve `'AP'`
- `altova:pad-string-right('AP', 3, 'YZ')` devuelve un error indicando que el carácter de relleno es demasiado largo.

repeat-string [altova:]

altova:repeat-string(CadenaEntrada *as xs:string*, **Repeticiones** *as xs:integer*) como **xs:string** **XP2** **XQ1** **XP3.1** **XQ3.1**

Genera una cadena que está compuesta por el primer argumento `CadenaEntrada` repetida tantas veces como indique el argumento `Repeticiones`.

Ejemplo

- **altova:repeat-string**("Altova #", 3)
  devuelve `Altova #Altova #Altova #`

substring-after-last [altova:]

altova:substring-after-last(CadenaPrincipal *as xs:string*, **CadenaPrueba** *as xs:string*) como **xs:string** **XP3.1** **XQ3.1**

Si **CadenaPrueba** se encuentra en **CadenaPrincipal**, la función devuelve la subcadena que aparece después de **CadenaPrueba** en **CadenaPrincipal**. Si **CadenaPrueba** no está en **CadenaPrincipal**, entonces devuelve la cadena vacía. Si **CadenaPrueba** es una cadena vacía, entonces devuelve la **CadenaPrincipal** entera. Si **CadenaPrueba** aparece varias veces en **CadenaPrincipal**, la función devuelve la subcadena que aparece después de la última **CadenaPrueba**.

Ejemplos

- **altova:substring-after-last**('ABCDEFGH', 'B') devuelve `'CDEFGH'`
- **altova:substring-after-last**('ABCDEFGH', 'BC') devuelve `'DEFGH'`
- **altova:substring-after-last**('ABCDEFGH', 'BD') devuelve `''`
- **altova:substring-after-last**('ABCDEFGH', 'Z') devuelve `''`
- **altova:substring-after-last**('ABCDEFGH', '') devuelve `'ABCDEFGH'`
- **altova:substring-after-last**('ABCD-ABCD', 'B') devuelve `'CD'`
- **altova:substring-after-last**('ABCD-ABCD-ABCD', 'BCD') devuelve `''`

substring-before-last [altova:]

altova:substring-before-last(CadenaPrincipal *as xs:string*, **CadenaPrueba** *as xs:string*) como **xs:string** **XP3.1** **XQ3.1**

Si **CadenaPrueba** se encuentra en **CadenaPrincipal**, la función devuelve la subcadena que aparece después de **CadenaPrueba** en **CadenaPrincipal**. Si **CadenaPrueba** no está en **CadenaPrincipal**, entonces devuelve la cadena vacía. Si **CadenaPrueba** es una cadena vacía, entonces devuelve la **CadenaPrincipal** entera. Si **CadenaPrueba** aparece varias veces en **CadenaPrincipal**, la función

 © 2017-2023 Altova GmbH

devuelve la subcadena que aparece antes de la última **CadenaPrueba**.

Ejemplos

- **altova:substring-before-last**('ABCDEFGH', 'B') devuelve 'A'
- **altova:substring-before-last**('ABCDEFGH', 'BC') devuelve 'A'
- **altova:substring-before-last**('ABCDEFGH', 'BD') devuelve ''
- **altova:substring-before-last**('ABCDEFGH', 'Z') devuelve ''
- **altova:substring-before-last**('ABCDEFGH', '') devuelve ''
- **altova:substring-before-last**('ABCD-ABCD', 'B') devuelve 'ABCD-A'
- **altova:substring-before-last**('ABCD-ABCD-ABCD', 'ABCD') devuelve 'ABCD-ABCD-'

substring-pos [altova:]

altova:substring-pos(Cadena *as xs:string*, **CadenaBúsqueda** *as xs:string***)** como **xs:integer** **XP3.1** **XQ3.1**

Devuelve la posición de carácter de la primera instancia de **CadenaBúsqueda** en **Cadena**. La posición de carácter se devuelve como número entero. El primer carácter de **CadenaBúsqueda** tiene la posición 1. Si **CadenaBúsqueda** no aparece dentro de **Cadena**, la función devuelve el entero 0. Para buscar la segunda instancia de **CadenaBúsqueda**, etc. use la otra firma de esta función.

Ejemplos

- **altova:substring-pos**('Altova', 'to') devuelve 3
- **altova:substring-pos**('Altova', 'tov') devuelve 3
- **altova:substring-pos**('Altova', 'tv') devuelve 0
- **altova:substring-pos**('AltovaAltova', 'to') devuelve 3

altova:substring-pos(Cadena *as xs:string*, **CadenaBúsqueda** *as xs:string*, **Entero** *as xs:integer***)** como **xs:integer** **XP3.1** **XQ3.1**

Devuelve la posición de carácter de **CadenaBúsqueda** en **Cadena**. La búsqueda de **CadenaBúsqueda** empieza en la posición de carácter dada por el argumento **Entero** (es decir, no se busca en la subcadena anterior a esta posición). El entero devuelto, sin embargo, es la posición que la cadena encontrada tiene en **Cadena**. Esta firma es muy práctica si quiere buscar la segunda posición, etc. de una cadena que aparece varias veces dentro de **Cadena**. Si **CadenaBúsqueda** no aparece en **Cadena**, la función devuelve el entero 0.

Ejemplos

- **altova:substring-pos**('Altova', 'to', 1) devuelve 3
- **altova:substring-pos**('Altova', 'to', 3) devuelve 3
- **altova:substring-pos**('Altova', 'to', 4) devuelve 0
- **altova:substring-pos**('Altova-Altova', 'to', 0) devuelve 3
- **altova:substring-pos**('Altova-Altova', 'to', 4) devuelve 10

substring-pos [altova:]

altova:substring-pos(Cadena *as xs:string*, **CadenaBúsqueda** *as xs:string***)** como **xs:integer** **XP3.1** **XQ3.1**

Devuelve la posición de carácter de la primera instancia de **CadenaBúsqueda** en **Cadena**. La posición de carácter se devuelve como número entero. El primer carácter de **CadenaBúsqueda** tiene la posición 1. Si **CadenaBúsqueda** no aparece dentro de **Cadena**, la función devuelve el entero 0. Para buscar la segunda instancia de **CadenaBúsqueda**, etc. use la otra firma de esta función.

Ejemplos

- `altova:substring-pos('Altova', 'to')` devuelve `3`
- `altova:substring-pos('Altova', 'tov')` devuelve `3`
- `altova:substring-pos('Altova', 'tv')` devuelve `0`
- `altova:substring-pos('AltovaAltova', 'to')` devuelve `3`

altova:substring-pos(Cadena *as xs:string*, **CadenaBúsqueda** *as xs:string*, **Entero** *as xs:integer*) como **xs:integer** XP3.1 XQ3.1
Devuelve la posición de carácter de **CadenaBúsqueda** en **Cadena**. La búsqueda de **CadenaBúsqueda** empieza en la posición de carácter dada por el argumento **Entero** (es decir, no se busca en la subcadena anterior a esta posición). El entero devuelto, sin embargo, es la posición que la cadena encontrada tiene en **Cadena**. Esta firma es muy práctica si quiere buscar la segunda posición, etc. de una cadena que aparece varias veces dentro de **Cadena**. Si **CadenaBúsqueda** no aparece en **Cadena**, la función devuelve el entero `0`.

Ejemplos

- `altova:substring-pos('Altova', 'to', 1)` devuelve `3`
- `altova:substring-pos('Altova', 'to', 3)` devuelve `3`
- `altova:substring-pos('Altova', 'to', 4)` devuelve `0`
- `altova:substring-pos('Altova-Altova', 'to', 0)` devuelve `3`
- `altova:substring-pos('Altova-Altova', 'to', 4)` devuelve `10`

trim-string [altova:]

altova:trim-string(CadenaEntrada *as xs:string*) como **xs:string** XP3.1 XQ3.1
Esta función toma un argumento `xs:string`, quita los espacios en blanco iniciales y finales y devuelve un `xs:string` "recortado".

Ejemplos

- `altova:trim-string("  Hello World  ")` devuelve `"Hello World"`
- `altova:trim-string("Hello World  ")` devuelve `"Hello World"`
- `altova:trim-string("  Hello World")` devuelve `"Hello World"`
- `altova:trim-string("Hello World")` devuelve `"Hello World"`
- `altova:trim-string("Hello  World")` devuelve `"Hello  World"`

trim-string-left [altova:]

altova:trim-string-left(CadenaEntrada *as xs:string*) como **xs:string** XP3.1 XQ3.1
Esta función toma un argumento `xs:string`, quita los espacios en blanco iniciales y devuelve un `xs:string` recortado por la izquierda.

Ejemplos

- `altova:trim-string-left("  Hello World  ")` devuelve `"Hello World  "`
- `altova:trim-string-left("Hello World  ")` devuelve `"Hello World  "`
- `altova:trim-string-left("  Hello World")` devuelve `"Hello World"`
- `altova:trim-string-left("Hello World")` devuelve `"Hello World"`
- `altova:trim-string-left("Hello  World")` devuelve `"Hello  World"`

© 2017-2023 Altova GmbH

▼ trim-string-right [altova:]

altova:trim-string-right(CadenaEntrada *as xs:string***)** como **xs:string** XP3.1 XQ3.1
Esta función toma un argumento `xs:string`, quita los espacios en blanco finales y devuelve una cadena `xs:string` recortada por la derecha.

Ejemplos

- **altova:trim-string-right**`("  Hello World  ")` devuelve `"  Hello World"`
- **altova:trim-string-right**`("Hello World  ")` devuelve `"Hello World"`
- **altova:trim-string-right**`("  Hello World")` devuelve `"  Hello World"`
- **altova:trim-string-right**`("Hello World")` devuelve `"Hello World"`
- **altova:trim-string-right**`("Hello  World")` devuelve `"Hello  World"`

10.2.1.9 Funciones XPath/XQuery varias

Estas funciones de extensión XPath/XQuery generales son compatibles con la versión actual de RaptorXML Server y se pueden usar en (i) expresiones XPath en contextos XSLT o (ii) en expresiones XQuery en documentos XQuery.

Nota sobre el nombre de las funciones y lenguajes

Puede utilizar todas las funciones de extensión de Altova en sus expresiones XPath/XQuery. Con ellas conseguirá funciones adicionales no disponibles en la biblioteca de funciones estándar de XPath, XQuery y XSLT. Las funciones de extensión de Altova están en el **espacio de nombres** **http://www.altova.com/xslt-extensions** y en esta sección se presentan con el prefijo, que se supone estará enlazado al espacio de nombres señalado. Tenga en cuenta que en futuras versiones del producto algunas funciones pueden dejar de ser compatibles o su comportamiento puede cambiar. Por tanto, consulte siempre la documentación del producto para conocer el funcionamiento de estas funciones en cada versión del producto.

Funciones XPath (en expresiones XPath en XSLT):	XP1 XP2 XP3.1.1
Funciones XSLT (en expresiones XPath en XSLT):	XSLT1 XSLT2 XSLT3
Funciones XQuery (en expresiones XQuery en XQuery):	XQ1 XQ3.1

▼ Clic para expandir/contraer

altova:decode-string(Input *as xs:base64Binary***) como xs:string** XP3.1 XQ3.1
altova:decode-string(Input *as xs:base64Binary***, Encoding como** *xs:string***) como xs:string** XP3.1 XQ3.1
Descifra la entrada en base64Binary en una cadena con el cifrado que se indique. Si no se indica ninguno se usa UTF-8. Estos son los cifrados compatibles: `US-ASCII, ISO-8859-1, UTF-16, UTF-16LE, UTF-16BE, ISO-10646-UCS2, UTF-32, UTF-32LE, UTF-32BE, ISO-10646-UCS4`

Ejemplos

- `altova:decode-string($XML1/MailData/Meta/b64B)` devuelve la entrada en base64Binary como cadena de texto cifrada en UTF-8
- `altova:decode-string($XML1/MailData/Meta/b64B, "UTF-8")` devuelve la entrada en base64Binary como cadena de texto cifrada en UTF-8
- `altova:decode-string($XML1/MailData/Meta/b64B, "ISO-8859-1")` devuelve la entrada en base64Binary como una cadena de texto cifrada en ISO-8859-1

encode-string [altova:]

altova:encode-string(InputString *como xs:string***) como xs:base64Binaryinteger** XP3.1 XQ3.1
altova:encode-string(InputString *como xs:string*, **Encoding** *como xs:string***) como xs:base64Binaryinteger** XP3.1 XQ3.1
Cifra una cadena de texto usando el cifrado que se indique. Si no se indica ninguno, entonces se usa UTF-8. La cadena cifrada se convierte en caracteres base64Binary y se devuelve el valor base64Binary convertido. De momento se admite UTF-8, pero ampliaremos la compatibilidad a: `US-ASCII, ISO-8859-1, UTF-16, UTF-16LE, UTF-16BE, ISO-10646-UCS2, UTF-32, UTF-32LE, UTF-32BE, ISO-10646-UCS4`

Ejemplos

- `altova:encode-string("Altova")` devuelve el equivalente en base64Binary de la cadena de texto cifrada en UTF-8 `"Altova"`
- `altova:encode-string("Altova", "UTF-8")` devuelve el equivalente en base64Binary de la cadena de texto cifrada en UTF-8 `"Altova"`

get-temp-folder [altova:]

altova:get-temp-folder() como `xs:string` XP2 XQ1 XP3.1 XQ3.1
Esta función no toma ningún argumento. Devuelve la ruta de acceso de la carpeta temporal del usuario actual.

Ejemplo

- `altova:get-temp-folder()`
  en un equipo Windows devuelve (más o menos) `C:\Usuarios\<Usuario>\AppData\Local\Temp\` como valor de tipo `xs:string`.

generate-guid [altova:]

altova:generate-guid() as xs:string XP2 XQ1 XP3.1 XQ3.1
Genera una cadena única de la interfaz gráfica del usuario.

Ejemplo

- `altova:generate-guid()` devuelve (por ejemplo) `85F971DA-17F3-4E4E-994E-99137873ACCD`

high-res-timer [altova:]

altova:high-res-timer() como **xs:double** XP3.1 XQ3.1
Devuelve un valor de temporizador de alta resolución en segundos. La presencia de un temporizador de

 © 2017-2023 Altova GmbH

alta resolución en un sistema permite hacer mediciones de alta precisión si es necesario (por ejemplo, en animaciones y para precisar de forma exacta horas de ejecución de código). Esta función ofrece la resolución del temporizador de alta resolución del sistema.

Ejemplos

- `altova:high-res-timer()` devuelve algo como `'1.16766146154566E6'`

parse-html [altova:]

**`altova:parse-html(HTMLText` *`as xs:string`*`)` como `node()` XP3.1 XQ3.1
El argumento `HTMLText` es una cadena que contiene el texto de un documento HTML. La función crea una estructura HTML a partir de la cadena. La cadena enviada puede contener o no el elemento HTML. En ambos casos el elemento raíz de la estructura es un elemento llamado **`HTML`**. Asegúrese de que el código HTML de la cadena enviada es válido.

Ejemplos

- `altova:parse-html("<html><head/><body><h1>Header</h1></body></html>")` crea una estructura HTML a partir de la cadena enviada

sleep[altova:]

`altova:sleep(Millisecs` *`como xs:integer`*`)` como `empty-sequence()` XP2 XQ1 XP3.1 XQ3.1
Suspende la ejecución de la operación actual durante el número de milisegundos dado por el argumento `Millisecs`.

Ejemplos

- `altova:sleep(1000)` suspende la ejecución de la operación actual durante 1000 millisegundos.

[Subir 493]

10.2.1.10 Funciones de extensión para gráficos

Las funciones para gráficos que aparecen a continuación sirven para crear, generar y guardar gráficos como imágenes. Esta funciones son compatibles con la versión actual de su producto de Altova. No obstante, tenga en cuenta que en futuras versiones del producto algunas funciones pueden dejar de ser compatibles o su comportamiento puede cambiar. Por tanto, consulte siempre la documentación del producto para conocer el funcionamiento de estas funciones en cada versión del producto.

Nota: las funciones para gráficos solamente son compatibles con los **productos servidor** de Altova y con las ediciones **Enterprise Edition** de los productos de Altova.

Nota: en los productos servidor de Altova los formatos de imagen compatibles para gráficos son `jpg`, `png` y `bmp`. La opción más recomendable es `png` porque no pierde información y es un formato comprimido. En las ediciones **Enterprise Edition** de las herramientas de escritorio de Altova, los formatos compatibles son `jpg, png, bmp` y `gif`.

Funciones para generar y guardar gráficos

Estas funciones toman el objeto del gráfico (obtenido con las funciones de creación de gráficos) y generan una imagen o guardan una imagen en un archivo.

altova:generate-chart-image ($chart, $width, $height, $encoding) as atomic

donde

- $chart es el componente de extensión de gráficos obtenido con la función create-chart
- $width y $height deben especificarse con una unidad de longitud
- $encoding puede ser x-binarytobase64 o x-binarytobase16

La función devuelve la imagen del gráfico en la codificación elegida.

altova:generate-chart-image ($chart, $width, $height, $encoding, $imagetype) as atomic

donde

- $chart es el componente de extensión de gráficos obtenido con la función create-chart
- $width y $height deben especificarse con una unidad de longitud
- $encoding puede ser x-binarytobase64 o x-binarytobase16
- $imagetype puede ser uno de estos formatos de imagen: png, gif, bmp, jpg, jpeg. Recuerde que el formato gif no es compatible con los productos servidor de Altova (*ver nota al principio de este apartado*)

La función devuelve la imagen del gráfico en la codificación y formato de imagen elegidos.

altova:save-chart-image ($chart, $filename, $width, $height) as empty() ***(solo en Windows)***

donde

- $chart es el componente de extensión de gráficos obtenido con la función create-chart
- $filename es la ruta de acceso del archivo y el nombre de archivo donde se debe guardar la imagen del gráfico
- $width y $height deben especificarse con una unidad de longitud

La función guarda la imagen del gráfico en el archivo especificado en $filename. También puede usar la función xsl:result-document con encoding="x-base64tobinary", donde el contenido de imagen-datos se obtiene con las funciones generate-chart-image() o chart().

altova:save-chart-image ($chart, $filename, $width, $height, $imagetype) as empty() ***(solo en Windows)***

donde

- $chart es el componente de extensión de gráficos obtenido con la función create-chart
- $filename es la ruta de acceso del archivo y el nombre de archivo donde se debe guardar la imagen del gráfico

 © 2017-2023 Altova GmbH

- `$width` y `$height` deben especificarse con una unidad de longitud
- `$imagetype` puede ser uno de estos formatos de imagen: `png`, `gif`, `bmp`, `jpg`, `jpeg`. Recuerde que el formato `gif` no es compatible con los productos servidor de Altova (*ver nota al principio de este apartado*)

La función guarda la imagen del gráfico en el archivo especificado en `$filename` en el formato de imagen elegido. También puede usar la función `xsl:result-document` con `encoding="x-base64tobinary"`, donde el contenido de imagen-datos se obtiene con las funciones `generate-chart-image()` o `chart()`.

Funciones para crear gráficos

Puede usar estas funciones para crear gráficos.

`altova:create-chart``($chart-config, $chart-data-series*) como componente de extensión de gráficos`

donde

- `$chart-config` es el componente de extensión chart-config obtenido con la función `create-chart-config` o con la función `create-chart-config-from-xml`
- `$chart-data-series` es el componente de extensión chart-data-series obtenido con la función `create-chart-data-series` o con la función `create-chart-data-series-from-rows`

La función devuelve un componente de extensión de gráficos, que se crea a partir de los datos suministrados con los argumentos.

`altova:chart``($chart-config, $chart-data-series*) como componente de extensión de gráficos`

donde

- `$chart-config` es el componente de extensión de gráficos. Se trata de una serie no ordenada de cuatro pares clave:valor en las que las claves son: `"width"`, `"height"`, `"title"`, and `"kind"`. Los valores de `width` y `height` son números enteros. El valor de `kind` es uno de estos: `Pie`, `Pie3d`, `BarChart`, `BarChart3d`, `BarChart3dGrouped`, `LineChart`, `ValueLineChart`, `RoundGauge`, `BarGauge`.
- Cada `$chart-data-series` es una matriz de tamaño 3 donde cada matriz define una serie chart-data. Cada matriz se compone de: (i) el nombre de la serie de datos, (ii) los valores del eje X y (iii) los valores del eje Y. Se pueden enviar varias series de datos; en el ejemplo siguiente, por ejemplo, las dos matrices dan datos, respectivamente, de las temperaturas mensuales máxima y mínima.

La función devuelve un elemento de tipo `xs:base64Binary` que contiene la imagen del gráfico. Esta imagen se crea a partir de los datos suministrados con los argumentos de la función. Recuerde que, al usar esta función matrices y asignaciones, solo se puede usar en XPath 3.1, XQuery 3.1 o XSLT 3.0.

Ejemplo:
```
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) como componente de extensión de gráficos`

donde

- `$type-name` indica el gráfico que se va a crear: `Pie`, `Pie3d`, `BarChart`, `BarChart3d`, `BarChart3dGrouped`, `LineChart`, `ValueLineChart`, `RoundGauge`, `BarGauge`
- `$title` es el nombre del gráfico

La función devuelve un componente de extensión chart-config que contiene los datos de configuración del gráfico.

altova:create-chart-config-from-xml(`$xml-struct`) como componente de extensión de gráficos chart-config

donde

- `$xml-struct` es la estructura XML que contiene los datos de configuración del gráfico

La función devuelve un componente de extensión chart-config que contiene los datos de configuración del gráfico. Estos datos se suministran en un fragmento de código XML 500.

altova:create-chart-data-series(`$series-name?, $x-values*, $y-values*`) como componente de extensión de gráficos chart-data-series

donde

- `$series-name` especifica el nombre de la serie
- `$x-values` presenta la lista de valores del eje X
- `$y-values` presenta la lista de valores del eje Y

La función devuelve un componente de extensión chart-data-series que contiene los datos necesarios para generar el gráfico: es decir, el nombre de las series y los datos de los ejes.

altova:create-chart-data-row(`x, y1, y2, y3, ...`) como componente de extensión de gráficos chart-data-x-*N*y-row

donde

- `x` es el valor de la columna del eje X de la fila de datos del gráfico
- `yN` son los valores de las columnas del eje Y

La función devuelve un componente de extensión chart-data-x-Ny-row, que contiene los datos para la columna del eje X y las columnas del eje Y de una sola serie.

altova:create-chart-data-series-from-rows(`$series-names as xs:string*, $row*`) como componente de extensión de gráficos chart-data-series

donde

- `$series-name` es el nombre de la series que se debe crear
- `$row` es el componente de extensión chart-data-x-Ny-row que se debe crear como serie

 © 2017-2023 Altova GmbH

La función devuelve un componente de extensión chart-data-series, que contiene los datos para el eje X y el eje Y de la serie.

altova:create-chart-layer($chart-config, $chart-data-series*) como componente de extensión de gráficos chart-layer

donde

- $chart-config es el componente de extensión chart-config obtenido con la función create-chart-config o con la función create-chart-config-from-xml
- $chart-data-series es el componente de extensión chart-data-series que se obtiene con la función create-chart-data-series o con la función create-chart-data-series-from-rows

La función devuelve un componente de extensión chart-layer, que contiene los datos de la capa de gráfico.

altova:create-multi-layer-chart($chart-config, $chart-data-series*, $chart-layer*)

donde

- $chart-config es el componente de extensión chart-config obtenido con la función create-chart-config o con la función create-chart-config-from-xml
- $chart-data-series es el componente de extensión chart-data-series obtenido con la función create-chart-data-series o con la función create-chart-data-series-from-rows
- $chart-layer es el componente de extensión de gráficos multicapas obtenido con la función create-chart-layer

La función devuelve un componente de extensión de gráficos multicapas.

altova:create-multi-layer-chart($chart-config, $chart-data-series*, $chart-layer*, xs:boolean $mergecategoryvalues)

donde

- $chart-config es el componente de extensión chart-config obtenido con la función create-chart-config o con la función create-chart-config-from-xml
- $chart-data-series es el componente de extensión chart-data-series obtenido con la función create-chart-data-series o con la función create-chart-data-series-from-rows
- $chart-layer es el componente de extensión de gráficos multicapa obtenido con la función create-chart-layer
- $mergecategoryvalues combina los valores de varias series de datos si true y no los combina si false

La función devuelve un componente de extensión de gráficos multicapa.

10.2.1.10.1 Estructura XML de los datos de gráficos

A continuación puede ver un fragmento de código XML con datos de gráfico, tal y como aparecería para las funciones de extensión para gráficos de Altova[495]. Esto afecta al aspecto del gráfico. No todos los elementos se utilizan para todos los tipos de gráfico. Por ejemplo, el elemento `<Pie>` se omite en los gráficos de barras.

Nota: las funciones para gráficos son compatibles solamente con las ediciones **Enterprise** y **Server**.

```
<chart-config>
        <General
        SettingsVersion="1"  debe darse
        ChartKind="BarChart"  Pie, Pie3d, BarChart, StackedBarChart, BarChart3d, BarChart3dGrouped,
LineChart, ValueLineChart, AreaChart, StackedAreaChart, RoundGauge, BarGauge, CandleStick
        BKColor="#ffffff"  Color
        BKColorGradientEnd="#ffffff"  Color. En caso de degradado, BKColor y BKColorGradientEnd
definen el color del degradado
        BKMode="#ffffff"  Solid, HorzGradient, VertGradient
        BKFile="Ruta+NombreArchivo"  Cadena. Si el archivo existe, su contenido se dibuja sobre el fondo.
        BKFileMode="Stretch"  Stretch, ZoomToFit, Center, Tile
        ShowBorder="1"  Bool
        PlotBorderColor="#000000"  Color
        PlotBKColor="#ffffff"  Color
        Title=""  Cadena de texto
        ShowLegend="1"  Bool
        OutsideMargin="3.%"  PorcentajeOPíxel
        TitleToPlotMargin="3.%"  PorcentajeOPíxel
        LegendToPlotMargin="3.%"  PorcentajeOPíxel
        Orientation="vert"  Enums. Los valores posibles son: vert, horz
        >
        <TitleFont
           Color="#000000"  Color
           Name="Tahoma"  Cadena de texto
           Bold="1"  Bool
           Italic="0"  Bool
           Underline="0"  Bool
           MinFontHeight="10.pt"  TamañoFuente (solo valores pt)
           Size="8.%"  TamañoFuente />
        <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"
```

 © 2017-2023 Altova GmbH

```
      MinFontHeight="10.pt"
      Size="5.%" />
</General>

<Line
      ConnectionShapeSize="1.%"  PorcentajeOPixel
      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 es totalmente transparente y 0 es opaco
      OutlineColor="#000000"  Color
      ShowOutline="1"  Bool
/>

<CandleStick
      FillHighClose="0"  Bool. Si es 0, el cuerpo está vacío. Si es 1, FillColorHighClose se usa para el cuerpo de la vela
      FillColorHighClose="#ffffff"  Color. Para el cuerpo de la vela cuando el valor de close > que el valor de open
      FillHighOpenWithSeriesColor="1"  Bool. Si es true, el color de la serie se usa para rellenar el cuerpo de la vela cuando el valor de open > que el valor de close
      FillColorHighOpen="#000000"  Color. Para el cuerpo de la vela cuando el valor de open > que el valor de close y FillHighOpenWithSeriesColor es false
/>

<Colors  Combinación de colores definida por el usuario. Este elemento está vacío por defecto, excepto el estilo, y no tiene atributos Color
      UseSubsequentColors ="1"  Booleano. Si es 0, entonces se superpone el color. Si es 1, se usan los colores siguientes de la capa de gráfico anterior
      Style="User"  Valores posibles: "Default", "Grayscale", "Colorful", "Pastel", "User"
      Colors="#52aca0"  Color: solamente se añade para el conjunto de colores definido por el usuario
      Colors1="#d3c15d"  Color: solamente se añade para el conjunto de colores definido por el usuario
      Colors2="#8971d8"  Color: solamente se añade para el conjunto de colores definido por el usuario
      ...
      ColorsN=""  Cada conjunto de colores puede tener un máximo de diez colores, de Colors a Colors9
</Colors>

<Pie
```

```
            ShowLabels="1"  Bool
            OutlineColor="#404040"  Color
            ShowOutline="1"  Bool
            StartAngle="0."  Double
            Clockwise="1"  Bool
            Draw2dHighlights="1"  Bool
            Transparency="0"  Int (De 0 a 255: 0 es opaco, 255 es totalmente transparente)
            DropShadowColor="#c0c0c0"  Color
            DropShadowSize="5.%"  PorcentajeOPíxel
            PieHeight="10.%"  PorcentajeOPíxel. Los valores de píxel pueden ser diferente en el
resultado debido a la inclinación 3D
            Tilt="40.0"  Double (De 10 a 90: la inclinación 3D en grados de un gráfico circular 3D)
            ShowDropShadow="1"  Bool
            ChartToLabelMargin="10.%"  PorcentajeOPíxel
            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: intervalo manual
                  RangeTill="1."  Double : intervalo manual
                  LabelToAxisMargin="3.%"  PorcentajeOPíxel
                  AxisLabel=""  Cadena de texto
                  AxisColor="#000000"  Color
                  AxisGridColor="#e6e6e6"  Color
                  ShowGrid="1"  Bool
                  UseAutoTick="1"  Bool
                  ManualTickInterval="1."  Double
                  AxisToChartMargin="0.px"  PorcentajeOPíxel
                  TickSize="3.px"  PorcentajeOPíxel
                  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  Eje (igual que 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: Si es false, XSize y YSize definen la relación de aspecto de los
ejes x e y. Si es true, la relación de aspecto es igual a la ventana del gráfico
        XSize="100.%"  PorcentajeOPíxel. Los valores en píxel pueden ser diferentes en el resultado
debido a la inclinación 3D y a la opción de ajustar al tamaño
        YSize="100.%"  PorcentajeOPíxel. Los valores en píxel pueden ser diferentes en el resultado
debido a la inclinación 3D y a la opción de ajustar al tamaño
        SeriesMargin="30.%"  PorcentajeOPíxel. Los valores en píxel pueden ser diferentes en el
resultado debido a la inclinación 3D y a la opción de ajustar al tamaño
        Tilt="20."  Double. De -90 a +90 grados
        Rot="20."  Double. De -359 a +359 grados
        FoV="50.">  Double. Campo de visión: de 1 a 120 grados
        >
        <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.%"  PorcentajeOPíxel
        MajorTickWidth="3.px"  PorcentajeOPíxel
        MajorTickLength="4.%"  PorcentajeOPíxel
        MinorTickWidth="1.px"  PorcentajeOPíxel
        MinorTickLength="3.%"  PorcentajeOPíxel
        BorderColor="#a0a0a0"  Color
        FillColor="#303535"  Color
        MajorTickColor="#a0c0b0"  Color
        MinorTickColor="#a0c0b0"  Color
        BorderWidth="2.%"  PorcentajeOPíxel
        NeedleBaseWidth="1.5%"  PorcentajeOPíxel
        NeedleBaseRadius="5.%"  PorcentajeOPíxel
        NeedleColor="#f00000"  Color
        NeedleBaseColor="#141414"  Color
        TickToTickValueMargin="5.%"  PorcentajeOPíxel
        MajorTickStep="10."  Double
        MinorTickStep="5."  Double
        RoundGaugeBorderToColorRange="0.%"  PorcentajeOPíxel
        RoundGaugeColorRangeWidth ="6.%"  PorcentajeOPíxel
        BarGaugeRadius="5.%"  PorcentajeOPíxel
        BarGaugeMaxHeight="20.%"  PorcentajeOPíxel
        RoundGaugeNeedleLength="45.%"  PorcentajeOPíxel
        BarGaugeNeedleLength="3.%"  PorcentajeOPíxel
        >
        <TicksFont
                Color="#a0c0b0"
                Name="Tahoma"
                Bold="0"
                Italic="0"
                Underline="0"
                MinFontHeight="10.pt"
                Size="4.%"
        />
```

 © 2017-2023 Altova GmbH

```
            <ColorRanges>   Intervalos de color definidos por el usuario. Está vacío por defecto y no
tiene elementos secundarios
                  <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 Ejemplo: funciones para gráficos

A continuación puede ver un ejemplo de documento XSLT que muestra cómo usar las funciones de extensión para gráficos de Altova[495]. Después ofrecemos un documento XML y una captura de pantalla de la imagen generada cuando el documento XML se procesa con el documento XSLT usando el motor XSLT 2.0.

Nota: las funciones para gráficos solamente son compatibles con las ediciones **Enterprise Edition** y **Server** de los productos de Altova.

Nota: para más información sobre las tablas de datos del gráfico, consulte la documentación de las herramientas XMLSpy y StyleVision de Altova.

Documento XSLT

Este documento XSLT usa las funciones de extensión para gráficos de Altova para generar un gráfico circular. Este XSLT se puede usar para procesar el documento XML que aparece más abajo.

```
<?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>
```


© 2017-2023 Altova GmbH

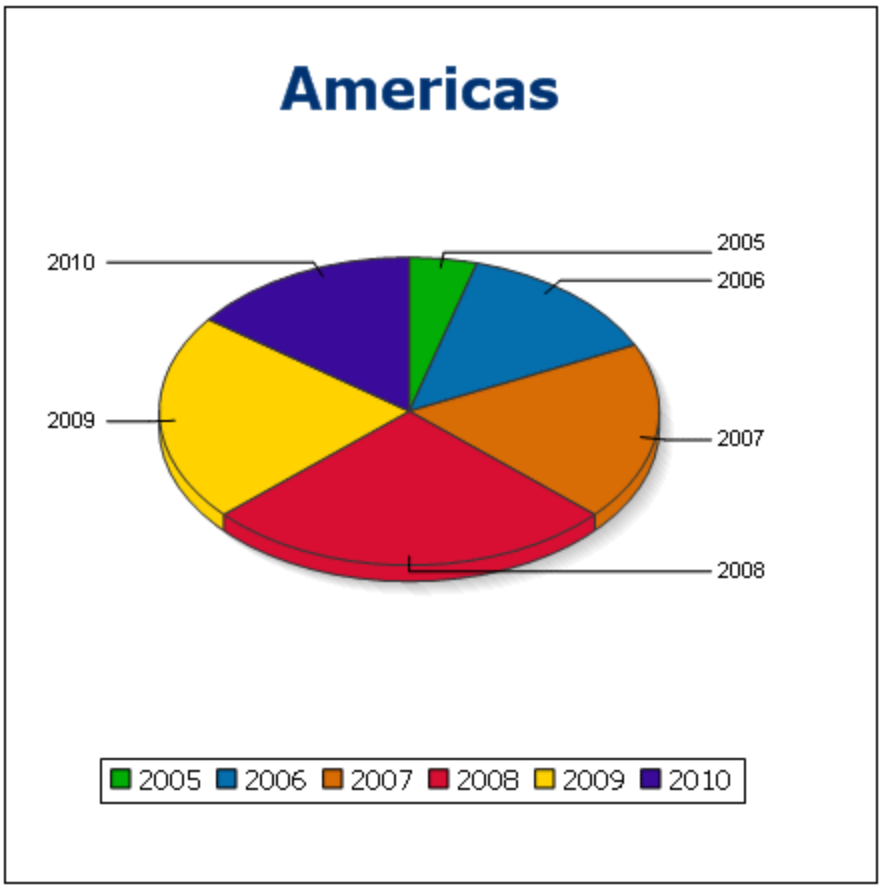
Documento XML

Este documento XML se puede procesar con el XSLT que aparece arriba. Los datos del documento XML se usan para generar el gráfico circular de la imagen que aparece más abajo.

```
<?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>
```

Imagen de salida

Este gráfico circular se generó al procesar el documento XML anterior con el archivo XSLT que aparece al principio de este apartado.



10.2.1.11 Funciones para códigos de barras

Los motores XSLT de Altova usan bibliotecas Java de terceros para crear códigos de barras. A continuación enumeramos las clases y los métodos públicos utilizados. Las clases se empaquetan en `AltovaBarcodeExtension.jar`, que está en la carpeta `<CarpetaArchivosPrograma>\Altova\Common2024\jar`.

Las bibliotecas Java utilizadas están en las subcarpetas de la carpeta `<CarpetaArchivosPrograma>\Altova\Common2024\jar`:

- `barcode4j\barcode4j.jar` (sitio web: http://barcode4j.sourceforge.net/)
- `zxing\core.jar` (sitio web: http://code.google.com/p/zxing/)

Los archivos de licencia están también en estas carpetas.

 © 2017-2023 Altova GmbH

Equipo virtual Java

Para poder usar las funciones de códigos de barras debe existir un equipo virtual en su equipo. A continuación se describe cómo encontrar la ruta de acceso a ese equipo.

- si está usando un producto de escritorio de Altova, la aplicación intentará detectar automáticamente la ruta de acceso al equipo virtual Java; para ello leerá (en este orden): (i) el registro de Windows y (ii) la variable de entorno **JAVA_HOME**. También puede añadir una ruta personal en el cuadro de diálogo "Opciones" de la aplicación; esta ruta tendrá prioridad frente a cualquier otra ruta de acceso a un equipo virtual Java que se detecte automáticamente.
- si está usando un producto servidor de Altova en un equipo Windows, la ruta de acceso al equipo virtual Java se leerá primero desde el registro de Windows; si esto no ocurre se usa la variable de entorno **JAVA_HOME**.
- si está usando un producto servidor de Altova en un equipo Linux o macOS, entonces asegúrese de que la variable de entorno **JAVA_HOME** está definida correctamente y la biblioteca Java de equipos virtuales (en Windows, el archivo **jvm.dll**) se encuentra en uno de estos directorios: **\bin\server** o **\bin\client**.

Ejemplo XSLT

A continuación puede ver un ejemplo de XSLT que ilustra el uso de funciones para códigos de barras en una hoja de estilos XSLT.

```
<?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="{altovaext:get-temp-folder()}barcode.png"/>
         </body>
      </html>
      <xsl:result-document
         href="{altovaext:get-temp-folder()}barcode.png"
         method="text" encoding="base64tobinary" >
         <xsl:variable name="barcodeObject"
            select="altovaext-barcode:newInstance(&apos;Code39&apos;,,string(&apos;some
value&apos;),
            96,0, (altovaext-barcode-property:new( &apos;setModuleWidth&apos;, 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>
```

Paquete `com.altova.extensions.barcode`

El paquete `com.altova.extensions.barcode` se utiliza para generar la mayoría de códigos de barras.

Se utilizan estas clases:

```
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`** *Utilizada para almacenar las propiedades del código de barras que se establecerán más tarde de forma dinámica*

```
    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`** *Registra la clase* `com.altova.extensions.barcode.proxy.zxing.QRCodeBean` *para el bean* `qrcode`*, además de las clases registradas por* `org.krysalis.barcode4j.DefaultBarcodeClassResolver`.

Ejemplo XQuery para generar códigos QR

A continuación puede ver un ejemplo de XQuery en el que se ven las funciones de código de barras que puede usar para generar una imagen de código QR.

```
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,:)
```

Paquete `com.altova.extensions.barcode.proxy.zxing`

El paquete `com.altova.extensions.barcode.proxy.zxing` se utiliza para generar códigos de barras de tipo QRCode.

Se utilizan estas clases:

© 2017-2023 Altova GmbH

```
class QRCodeBean
```

- *Extiende* `org.krysalis.barcode4j.impl.AbstractBarcodeBean`
- *Crea una interfaz* `AbstractBarcodeBean` *para* `com.google.zxing.qrcode.encoder`

```
    void generateBarcode(CanvasProvider canvasImp, String msg)
    void setQRErrorCorrectionLevel(QRCodeErrorCorrectionLevel level)
    BarcodeDimension calcDimensions(String msg)
    double getVerticalQuietZone()
    double getBarWidth()
```

```
class QRCodeErrorCorrectionLevel Nivel de corrección de errores para QRCode
    static QRCodeErrorCorrectionLevel byName(String name)
    "L" = ~7% correction
    "M" = ~15% correction
    "H" = ~25% correction
    "Q" = ~30% correction
```

10.2.2 Funciones de extensión varias

Los lenguajes de programación como Java y C# ofrecen varias funciones predefinidas que no están disponibles como funciones XQuery/XPath ni XSLT. Un ejemplo son las funciones matemáticas de Java `sin()` y `cos()`. Si los diseñadores de hojas de estilos XSLT y consultas XQuery tuvieran acceso a estas funciones, el área de aplicación de sus hojas de estilos y consultas aumentaría y su trabajo sería un poco más sencillo.

Los motores XSLT y XQuery de los productos de Altova admiten el uso de funciones de extensión en Java 512 y .NET 521, así como scripts MSXSL para XSLT 527.

Esta sección describe cómo usar funciones de extensión y scripts MSXSL en hojas de estilos XSLT y documentos XQuery. Las funciones de extensión pueden organizarse en varios grupos:

- Funciones de extensión Java 512
- Funciones de extensión .NET 521
- Scripts MSXSL para XSLT 527

En los apartados de esta sección nos ocupamos de tres aspectos fundamentales: (i) cómo se llaman las funciones en sus respectivas bibliotecas, (ii) qué reglas deben seguirse para convertir los argumentos de una llamada a función en el formato de entrada necesario de la función y (iii) qué reglas deben seguirse para la conversión del tipo devuelto.

Requisitos

Para que estas funciones de extensión funcionen es necesario tener Java Runtime Environment (para las funciones Java) y .NET Framework 2.0 o superior (para las funciones .NET) instalado en el equipo que ejecuta la transformación XSLT o XQuery.

10.2.2.1 Funciones de extensión Java

Puede usar una función de extensión Java dentro de una expresión XPath o XQuery para invocar un constructor Java o llamar a un método Java (estático o de instancia).

Un campo de una clase Java se trata como un método sin argumentos. Un campo puede ser estático o de instancia. Más adelante describimos cómo se accede a los campos estáticos y de instancia.

Este apartado tiene varias partes:

- Constructores Java 517
- Métodos estáticos y campos estáticos 518
- Métodos de instancia y campos de instancia 519
- Tipos de datos: conversión de XPath/XQuery en Java 519
- Tipos de datos: conversión de Java en XPath/XQuery 520

Tenga en cuenta que:

- si está usando un producto de escritorio de Altova, la aplicación intentará detectar automáticamente la ruta de acceso al equipo virtual Java; para ello leerá (en este orden): (i) el registro de Windows y (ii) la variable de entorno **JAVA_HOME**. También puede añadir una ruta personal en el cuadro de diálogo "Opciones" de la aplicación; esta ruta tendrá prioridad frente a cualquier otra ruta de acceso a un equipo virtual Java que se detecte automáticamente.
- si está usando un producto servidor de Altova en un equipo Windows, la ruta de acceso al equipo virtual Java se leerá primero desde el registro de Windows; si esto no ocurre se usa la variable de entorno **JAVA_HOME**.
- si está usando un producto servidor de Altova en un equipo Linux o macOS, entonces asegúrese de que la variable de entorno **JAVA_HOME** está definida correctamente y la biblioteca Java de equipos virtuales (en Windows, el archivo **jvm.dll**) se encuentra en uno de estos directorios: **\bin\server** o **\bin\client**.

Formato de la función de extensión

La función de extensión de la expresión XPath/XQuery debe tener este formato `prefijo:nombreFunción()`.

- La parte `prefijo:` identifica la función de extensión como función Java. Lo hace asociando la función de extensión con una declaración de espacio de nombres del ámbito, cuyo URI debe empezar por `java:` (*ver ejemplos más abajo*). La declaración de espacio de nombres debe identificar una clase Java, por ejemplo: `xmlns:myns="java:java.lang.Math"`. Sin embargo, también puede ser simplemente: `xmlns:myns="java"` (sin los dos puntos), dejando la identificación de la clase Java a la parte `nombreFunción()` de la función de extensión.
- La parte `nombreFunción()` identifica el método Java al que se llama y presenta los argumentos para el método (*ver ejemplos más abajo*). Sin embargo, si el URI de espacio de nombres identificado por la parte `prefijo:` no identifica una clase Java (*ver punto anterior*), entonces la clase Java debe identificarse en la parte `nombreFunción()`, antes de la clase y separada de la clase por un punto (*ver el segundo ejemplo XSLT que aparece más abajo*).

Nota: la clase a la que se llama debe estar en la ruta de acceso de clase del equipo.

© 2017-2023 Altova GmbH

Ejemplo de código XSLT

Aquí ofrecemos dos ejemplos de cómo se puede llamar a un método estático. En el primer ejemplo, el nombre de la clase (`java.lang.Math`) se incluye en el URI de espacio de nombres y, por tanto, no puede estar en la parte `nombreFunción()`. En el segundo ejemplo, la parte `prefijo:` presenta el prefijo `java:` mientras que la parte `nombreFunción()` identifica la clase y el método.

```
<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)" />
```

El método nombrado en la función de extensión (`cos()`) debe coincidir con el nombre de un método estático público de la clase Java nombrada (`java.lang.Math`).

Ejemplo de código XQuery

Aquí puede ver un ejemplo de código XQuery similar al código XSLT anterior:

```
<cosine xmlns:jMath="java:java.lang.Math">
   {jMath:cos(3.14)}
</cosine>
```

Clases Java definidas por el usuario

Si creó sus propias clases Java, a los métodos de estas clases se les llama de otra manera, dependiendo de: (i) si a las clases se accede por medio de un archivo JAR o de un archivo de clases y (ii) si estos archivos están en el directorio actual (el directorio del documento XSLT o XQuery). Para más información consulte los apartados Archivos de clases definidos por el usuario 513 y Archivos Jar definidos por el usuario 516. Recuerde que debe especificar las rutas de acceso de los archivos de clases que no están en el directorio actual y de todos los archivos JAR.

10.2.2.1.1 Archivos de clases definidos por el usuario

Si se accede a las clases por medio de un archivo de clases, entonces hay cuatro posibilidades:

- El archivo de clases está en un paquete. El archivo XSLT/XQuery está en la misma carpeta que el paquete Java. (*ver ejemplo* 514.)
- El archivo de clases no está en un paquete. El archivo XSLT/XQuery está en la misma carpeta que el archivo de clases. (*ver ejemplo* 514.)
- El archivo de clases está en un paquete. El archivo XSLT/XQuery está en una carpeta cualquiera. (*ver ejemplo* 515.)
- El archivo de clases no está en un paquete. El archivo XSLT/XQuery está una carpeta cualquiera. (*ver ejemplo* 515.)

Imaginemos que tenemos un archivo de clases que no está en un paquete y que está en la misma carpeta que el documento XSLT/XQuery. En este caso, puesto que en la carpeta se encuentran todas las clases, no es necesario especificar la ubicación del archivo. La sintaxis que se utiliza para identificar una clase es esta:

```
java:nombreClase
```

donde

`java`: indica que se está llamando a una función definida por el usuario (por defecto se cargan las clases Java del directorio actual)

`nombreClase` es el nombre de la clase del método elegido

La clase se identifica en un URI de espacio de nombres y el espacio de nombres se usa como prefijo para la llamada al método.

El archivo de clases está en un paquete. El archivo XSLT/XQuery está en la misma carpeta que el paquete Java

El código que aparece a continuación llama al método `getVehicleType()` de la clase `Car` del paquete `com.altova.extfunc`. El paquete `com.altova.extfunc` está en la carpeta `JavaProject`. El archivo XSLT también está en la carpeta `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>
```

El archivo de clases está referenciado. El archivo XSLT/XQuery está en la misma carpeta que el archivo de clases

El código que aparece a continuación llama al método `getVehicleType()` de la clase `Car`. Digamos que: (i) el archivo de clases `Car` está en esta carpeta: `JavaProject/com/altova/extfunc` y que (ii) esa carpeta es la del ejemplo siguiente. El archivo XSLT también está en la carpeta `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"/>
```

 © 2017-2023 Altova GmbH

```
<xsl:template match="/">
   <a>
   <xsl:value-of select="car:getVehicleType()"/>
   </a>
</xsl:template>

</xsl:stylesheet>
```

El archivo de clases está en un paquete. El archivo XSLT/XQuery está en una carpeta cualquiera

El código que aparece a continuación llama al método `getCarColor()` de la clase `Car` del paquete `com.altova.extfunc`. El paquete `com.altova.extfunc` está en la carpeta `JavaProject`. El archivo XSLT está en otra carpeta cualquiera. En este caso debe especificarse la ubicación del paquete dentro del URI como una cadena de consulta. La sintaxis es esta:

```
java:nombreClase[?ruta=uri-del-paquete]
```

donde

`java`: indica que se está llamando a una función Java definida por el usuario
`uri-del-paquete` es el URI del paquete Java
`nombreClase` es el nombre de la clase del método elegido

La clase se identifica en un URI de espacio de nombres y el espacio de nombres se usa como prefijo para la llamada al método. El ejemplo de código que aparece a continuación explica cómo se accede a un archivo de clases que está ubicado en un directorio que no es el directorio actual.

```
<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>
```

El archivo de clases no está en un paquete. El archivo XSLT/XQuery está una carpeta cualquiera

El código que aparece a continuación llama al método `getCarColor()` de la clase `Car`. Digamos que el archivo de clases `Car` está en la carpeta `C:/JavaProject/com/altova/extfunc` y que el archivo XSLT está en otra

carpeta cualquiera. En este caso debe especificarse la ubicación del paquete dentro del URI como una cadena de consulta. La sintaxis es esta:

```
java:nombreClase[?ruta=<uri-del-archivoClases>]
```

donde

`java`: indica que se está llamando a una función Java definida por el usuario
`uri-del-archivoClases` es el URI de la carpeta donde se ubica el archivo de clases
`nombreClase` es el nombre de la clase del método elegido

La clase se identifica en un URI de espacio de nombres y el espacio de nombres se usa como prefijo para la llamada al método. El ejemplo de código que aparece a continuación explica cómo se accede a un archivo de clases que está ubicado en un directorio que no es el directorio actual.

```
<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>
```

Nota: cuando se presenta una ruta de acceso por medio de una función de extensión, la ruta de acceso se añade al ClassLoader.

10.2.2.1.2 Archivos JAR definidos por el usuario

Si se accede a las clases por medio de un archivo JAR, entonces se debe especificar el URI del archivo JAR usando esta sintaxis:

```
xmlns:claseEspacioNombres="java:nombreClase?ruta=jar:uri-del-archivoJar!/"
```

Para la llamada al método se usa el prefijo del URI de espacio de nombres que identifica la clase: `claseEspacioNombres:método()`

En la sintaxis anterior:

`java`: indica que se está llamando a una función de Java
`nombreClase` es el nombre de la clase definida por el usuario
`?` es el separador entre el nombre de la clase y la ruta de acceso
`ruta=jar`: indica que se ofrece una ruta de acceso a un archivo JAR
`uri-del-archivoJar` es el URI del archivo JAR

 © 2017-2023 Altova GmbH

`!/` es el delimitador final de la ruta de acceso
`claseEspacioNombres:método()` es la llamada al método

Otra opción es dar el nombre de la clase con la llamada al método. Por ejemplo:

```
              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()
```

Y aquí puede ver un ejemplo de XSLT que usa un archivo JAR para llamar a una función de extensión Java:

```
<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>
```

Nota: cuando se presenta una ruta de acceso por medio de una función de extensión, la ruta de acceso se añade al ClassLoader.

10.2.2.1.3 Constructores

Una función de extensión se puede usar para llamar a un constructor Java. A todos los constructores se les llama con la pseudofunción `new()`.

Si el resultado de una llamada a un constructor Java se puede convertir de manera implícita a tipos de datos XPath/XQuery[520], entonces la llamada a la función de extensión Java devuelve una secuencia que es un tipo de datos XPath/XQuery. Si el resultado de una llamada a un constructor Java no se puede convertir a un tipo de datos XPath/XQuery adecuado, entonces el constructor crea un objeto Java contenido con un tipo que es el nombre de la clase que devuelve ese objeto Java. Por ejemplo, si se llama a un constructor para la clase `java.util.Date` (`java.util.Date.new()`), entonces se devuelve un objeto que tiene el tipo `java.util.Date`. Puede que el formato léxico del objeto devuelto no coincida con el formato léxico de un tipo de datos XPath y, por tanto, su valor debe convertirse al formato léxico del tipo de datos XPath pertinente y después al tipo de datos XPath.

Puede hacer dos cosas con el objeto Java creado por un constructor:

- Puede asignar el objeto a una variable:
```
<xsl:variable name="currentdate" select="date:new()"
xmlns:date="java:java.util.Date" />
```
- Puede pasar el objeto a una función de extensión (*ver* *métodos de instancia y campos de instancia* 519):
```
<xsl:value-of select="date:toString(date:new())" xmlns:date="java:java.util.Date" />
```

10.2.2.1.4 Métodos estáticos y campos estáticos

La llamada a un método estático la hace directamente su nombre Java y se hace presentando los argumentos para el método. A los campos estáticos (es decir, los métodos que no toman argumentos), como los campos de valor constante `E` y `PI`, se accede sin especificar ningún argumento.

Ejemplos de código XSLT

Aquí puede ver varios ejemplos de cómo se llama a métodos y campos estáticos:

```
<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)" />
```

Observe que las funciones de extensión anteriores tienen el formato `prefijo:nombreFunción()`. En los tres ejemplos anteriores, el prefijo es `jMath:`, que está asociado al URI de espacio de nombres `java:java.lang.Math`. (El URI de espacio de nombres debe empezar por `java:`. En los ejemplos anteriores se extiende para contener el nombre de la clase (`java.lang.Math`).) La parte `nombreFunción()` de las funciones de extensión debe coincidir con el nombre de una clase pública (p. ej. `java.lang.Math`) seguido del nombre de un método estático público con sus argumentos (como `cos(3.14)`) o de un campo estático público (como `PI()`).

En los tres ejemplos anteriores, el nombre de la clase se incluyó en el URI de espacio de nombres. Si no estuviera en el URI de espacio de nombres, se incluiría en la parte `nombreFunción()` de la función de extensión. Por ejemplo:

```
<xsl:value-of xmlns:java="java:"
              select="java:java.lang.Math.cos(3.14)" />
```

Ejemplo de XQuery

Un ejemplo de XQuery similar sería:

```
<cosine xmlns:jMath="java:java.lang.Math">
   {jMath:cos(3.14)}
</cosine>
```

 © 2017-2023 Altova GmbH

10.2.2.1.5 Métodos de instancia y campos de instancia

A un método de instancia se le pasa un objeto Java como primer argumento de la llamada a método. Dicho objeto Java suele crearse usando una función de extensión (por ejemplo, una llamada a un constructor) o un parámetro o una variable de hoja de estilos. Un ejemplo de código XSLT de este tipo sería:

```
<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>
```

En el ejemplo anterior el valor del nodo `enrollment/@type` se crea de la siguiente manera:

1. Se crea un objeto con un constructor para la clase `java.util.Date` (con el constructor `date:new()`).
2. Este objeto Java se pasa como argumento del método `jlang.Object.getClass`.
3. El objeto que obtiene el método `getClass` se pasa como argumento al método `jlang.Object.toString`.

El resultado (el valor de `@type`) será una cadena con este valor: `java.util.Date`.

En teoría, un campo de instancia es diferente de un método de instancia porque al campo de instancia no se pasa como argumento un objeto Java propiamente dicho. En su lugar se pasa como argumento un parámetro o variable. Sin embargo, el parámetro o la variable puede contener el valor devuelto por un objeto Java. Por ejemplo, el parámetro `CurrentDate` toma el valor que devolvió un constructor para la clase `java.util.Date`. Este valor se pasa después como argumento al método de instancia `date:toString` a fin de suministrar el valor de `/enrollment/@date`.

10.2.2.1.6 Tipos de datos: conversión de XPath/XQuery en Java

Cuando se llama a una función Java desde dentro de una expresión XPath/XQuery, el tipo de datos de los argumentos de la función es importante a la hora de determinar a cuál de las clases Java que tienen el mismo nombre se llama.

En Java se siguen estas reglas:

- Si hay más de un método Java con el mismo nombre, pero cada método tiene un número diferente de argumentos, entonces se selecciona el método Java que mejor se ajusta al número de argumentos de la llamada a función.
- Los tipos de datos de cadena, numéricos y booleanos de XPath/XQuery (*ver lista más abajo*) se convierten de forma implícita en el tipo de datos Java correspondiente. Si el tipo XPath/XQuery suministrado se puede convertir a más de un tipo Java (p. ej. `xs:integer`), entonces se selecciona el

tipo Java que se declaró para el método seleccionado. Por ejemplo, si el método Java al que se llama es `fx(decimal)` y el tipo de datos XPath/XQuery suministrado es `xs:integer`, entonces `xs:integer` se convierte en el tipo de datos Java `decimal`.

La tabla que aparece a continuación enumera las conversiones implícitas de los tipos de cadena, numéricos y booleanos XPath/XQuery en tipos de datos Java.

`xs:string`	`java.lang.String`
`xs:boolean`	`boolean` (primitivo), `java.lang.Boolean`
`xs:integer`	int, long, short, byte, float, double y sus clases contenedoras, como `java.lang.Integer`
`xs:float`	`float` (primitivo), `java.lang.Float`, `double` (primitivo)
`xs:double`	`double` (primitivo), `java.lang.Double`
`xs:decimal`	`float` (primitivo), `java.lang.Float`, `double`(primitivo), `java.lang.Double`

Los subtipos de los tipos de datos XML Schema de la tabla anterior (que se usan en XPath y XQuery) también se convierten en los tipos Java correspondientes al tipo antecesor del subtipo.

En algunos casos quizás no sea posible seleccionar el método Java correcto usando la información dada. Por ejemplo, imagine que:

- El argumento presentado es un valor `xs:untypedAtomic` de 10 y está destinado al método `mimétodo(float)`.
- Sin embargo, hay otro método en la clase que toma un argumento de otro tipo de datos: `mimétodo(double)`.
- Puesto que los métodos tienen el mismo nombre y el tipo suministrado (`xs:untypedAtomic`) se puede convertir correctamente tanto en `float` como en `double`, es posible que `xs:untypedAtomic` se convierta en `double` en lugar de en `float`.
- Por consiguiente, el método seleccionado no será el método necesario y quizás no produzca el resultado esperado. Una solución es crear un método definido por el usuario con un nombre diferente y usar ese método.

Los tipos que no aparecen en la lista anterior (p. ej. `xs:date`) no se convertirán y generarán un error. No obstante, tenga en cuenta que en algunos casos, es posible crear el tipo Java necesario usando un constructor Java.

10.2.2.1.7 Tipos de datos: conversión de Java en XPath/XQuery

Cuando un método Java devuelve un valor y el tipo de datos del valor es un tipo de cadena, numérico o booleano, entonces se convierte en el tipo de datos XPath/XQuery correspondiente. Por ejemplo, los tipos de datos Java `java.lang.Boolean` y `boolean` se convierten en `xsd:boolean`.

Las matrices unidimensionales devueltas por las funciones se extienden en una secuencia. Las matrices multidimensionales no se convierten y, por tanto, deberían ser contenidas.

 © 2017-2023 Altova GmbH

Cuando se devuelve un objeto Java contenido o un tipo de datos que no es de cadena, numérico ni booleano, puede garantizar la conversión del tipo XPath/XQuery necesario usando primero un método Java (p. ej. `toString`) para convertir el objeto Java en una cadena. En XPath/XQuery la cadena se puede modificar para ajustarse a la representación léxica del tipo necesario y convertirse después en dicho tipo (usando la expresión `cast as`, por ejemplo).

10.2.2.2 Funciones de extensión .NET

Si trabaja en la plataforma .NET desde un equipo Windows, puede usar funciones de extensión escritas en cualquier lenguaje .NET (p. ej. C#). Una función de extensión .NET se puede usar dentro de una expresión XPath/XQuery para invocar un constructor, una propiedad o un método (estático o de instancia) de una clase .NET.

A una propiedad de una clase .NET se le llama usando la sintaxis `get_NombrePropiedad()`.

Este apartado tiene varias partes:

- Constructores 523
- Métodos estáticos y campos estáticos 524
- Métodos de instancia y campos de instancia 525
- Tipos de datos: conversión de XPath/XQuery en .NET 526
- Tipos de datos: conversión de .NET en XPath/XQuery 527

Formato de la función de extensión

La función de extensión de la expresión XPath/XQuery debe tener este formato `prefijo:nombreFunción()`.

- La parte `prefijo:` está asociada a un URI que identifica la clase .NET.
- La parte `nombreFunción()` identifica el constructor, la propiedad o el método (estático o de instancia) dentro de la clase .NET y, si es necesario, suministra los argumentos.
- El URI debe empezar por `clitype:` (que identifica la función como función de extensión .NET).
- El formato `prefijo:nombreFunción()` de la función de extensión se puede usar con clases del sistema y con clases de un ensamblado cargado. No obstante, si se tiene que cargar una clase, será necesario suministrar parámetros que contengan la información necesaria.

Parámetros

Para cargar un ensamblado se usan estos parámetros:

`asm`	El nombre del ensamblado que se debe cargar.
`ver`	El número de versión (máximo cuatro enteros separados por puntos).
`sn`	El símbolo de clave del nombre seguro del ensamblado (16 dígitos hexadecimales).

`from`	Un URI que da la ubicación del ensamblado (DLL) que se debe cargar. Si el URI es relativo, es relativo al archivo XSLT o XQuery. Si está presente este parámetro, se ignoran los demás parámetros.
`partialname`	El nombre parcial del ensamblado. Se suministra a `Assembly.LoadWith.PartialName()`, que intentará cargar el ensamblado. Si está presente el parámetro `partialname`, se ignoran los demás parámetros.
`loc`	La configuración regional, por ejemplo, `en-US`. La configuración predeterminada es `neutral`.

Si el ensamblado se debe cargar desde un archivo DLL, use el parámetro `from` y omita el parámetro `sn`. Si el ensamblado se debe cargar desde el caché general de ensamblados (GAC), use el parámetro `sn` y omita el parámetro `from`.

Debe insertar un signo de interrogación final antes del primer parámetro y los parámetros deben separarse con un punto y coma (;). El nombre de parámetro da su valor con un signo igual (=), como en el ejemplo que aparece más abajo.

Ejemplos de declaraciones de espacios de nombres

Esto es un ejemplo de una declaración de espacio de nombres en XSLT que identifica la clase del sistema `System.Environment`:

```
xmlns:myns="clitype:System.Environment"
```

Esto es un ejemplo de una declaración de espacio de nombres en XSLT que identifica la clase que se debe cargar como `Trade.Forward.Scrip`:

```
xmlns:myns="clitype:Trade.Forward.Scrip?asm=forward;version=10.6.2.1"
```

Esto es un ejemplo de una declaración de espacio de nombres en XQuery que identifica la clase del sistema `MyManagedDLL.testClass`:. Existen dos tipos de clases:

1. Cuando el ensamblado se carga desde el GAC:
   ```
   declare namespace cs="clitype:MyManagedDLL.testClass?asm=MyManagedDLL;
          ver=1.2.3.4;loc=neutral;sn=b9f091b72dccfba8";
   ```
2. Cuando el ensamblado se carga desde el archivo DLL (ver las referencias parciales y completas):
   ```
          declare namespace cs="clitype:MyManagedDLL.testClass?from=file:///C:/Altova
          Projects/extFunctions/MyManagedDLL.dll;

   declare namespace cs="clitype:MyManagedDLL.testClass?from=MyManagedDLL.dll;
   ```

Ejemplo de código XSLT

Aquí puede ver un ejemplo de código XSLT que llama a funciones de la clase del sistema `System.Math`:

```
<xsl:stylesheet version="2.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
```

 © 2017-2023 Altova GmbH

```
   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>
```

La declaración de espacio de nombres del elemento `math` asocia el prefijo `math:` al URI `clitype:System.Math`. La parte inicial `clitype:` del URI indica que lo que sigue identifica una clase del sistema o una clase cargada. El prefijo `math:` de las expresiones XPath asocia las funciones de extensión al URI (y, por extensión, a la clase) `System.Math`. Las funciones de extensión identifican métodos en la clase `System.Math` y presenta argumentos cuando es necesario.

Ejemplo de código XQuery

Aquí puede ver un fragmento de código XQuery similar al ejemplo anterior:

```
<math xmlns:math="clitype:System.Math">
   {math:Sqrt(9)}
</math>
```

Tal y como ocurre con el código XSLT anterior, la declaración de espacio de nombres identifica la clase .NET, en este caso una clase del sistema. La expresión XQuery identifica el método al que se debe llamar y presenta el argumento.

10.2.2.2.1 Constructores

Una función de extensión se puede usar para llamar a un constructor .NET. A todos los constructores se les llama con la pseudofunción `new()`. Si hay más de un constructor para una clase, entonces se selecciona el constructor que más se ajusta al número de argumentos suministrados. Si no se encuentra ningún constructor que coincida con los argumentos suministrados, entonces se genera el error "`No constructor found`".

Constructores que devuelven tipos de datos XPath/XQuery

Si el resultado de una llamada a un constructor .NET se puede convertir de forma implícita en tipos de datos XPath/XQuery 520, entonces la función de extensión .NET devuelve una secuencia que es un tipo de datos XPath/XQuery.

Constructores que devuelven objetos .NET

Si el resultado de una llamada a un constructor .NET no se puede convertir a un tipo de datos XPath/XQuery adecuado, entonces el constructr crea un objeto .NET contenido con un tipo que es el nombre de la clase que

devuelve dicho objeto. Por ejemplo, si se llama al constructor para la clase `System.DateTime` (con `System.DateTime.new()`), entonces se devuelve un objeto que tiene un tipo `System.DateTime`.

Puede que el formato léxico del objeto devuelto no coincida con el formato léxico de un tipo de datos XPath. En estos casos, el valor devuelto (i) debe convertirse al formato léxico del tipo de datos XPath pertinente y (ii) debe convertirse en el tipo de datos XPath necesario.

Se pueden hacer tres cosas con un objeto .NET creado con un constructor:

- Se puede usar dentro de una variable:
```
<xsl:variable name="currentdate" select="date:new(2008, 4, 29)"
xmlns:date="clitype:System.DateTime" />
```
- Se puede pasar a una función de extensión (ver *Métodos de instancia y campos de instancia* 519):
```
<xsl:value-of select="date:ToString(date:new(2008, 4, 29))"
xmlns:date="clitype:System.DateTime" />
```
- Se puede convertir en un tipo de cadena, numérico o booleano:
```
<xsl:value-of select="xs:integer(date:get_Month(date:new(2008, 4, 29)))"
xmlns:date="clitype:System.DateTime" />
```

10.2.2.2.2 Metodos estáticos y campos estáticos

La llamada a un método estático la hace directamente su nombre y se hace presentando los argumentos para el método. El nombre usado en la llamada debe ser el mismo que un método estático público de la clase especificada. Si el nombre del método y el número de argumentos que se dio en la llamada a función coincide con algún método de la clase, entonces los tipos de los argumentos presentados se evalúan para encontrar el resultado ideal. Si no se encuentra ninguna coincidencia, se emite un error.

Nota: un campo de una clase .NET se trata como si fuera un método sin argumentos. Para llamar a una propiedad se usa la sintaxis `get_nombrePropiedad()`.

Ejemplos

Este ejemplo de código XSLT muestra una llamada a un método con un argumento (`System.Math.Sin(arg)`):

```
<xsl:value-of select="math:Sin(30)" xmlns:math="clitype:System.Math"/>
```

Este ejemplo de código XSLT muestra una llamada a un campo (que se trata como si fuera un método sin argumentos) (`System.Double.MaxValue()`):

```
<xsl:value-of select="double:MaxValue()" xmlns:double="clitype:System.Double"/>
```

Este ejemplo de código XSLT muestra una llamada a una propiedad (la sintaxis es `get_nombrePropiedad()`) (`System.String()`):

```
<xsl:value-of select="string:get_Length('my string')"
xmlns:string="clitype:System.String"/>
```

 © 2017-2023 Altova GmbH

Este ejemplo de código XQuery muestra una llamada a un método con un argumento (`System.Math.Sin(arg)`):

```
<sin xmlns:math="clitype:System.Math">
   { math:Sin(30) }
</sin>
```

10.2.2.2.3 Métodos de instancia y campos de instancia

Un método de instancia es un método al que se le pasa un objeto .NET como primer argumento de la llamada al método. Este objeto .NET se suele crear usando una función de extensión (por ejemplo, una llamada a un constructor) o un parámetro o una variable de una hoja de estilos. Un ejemplo de código XSLT para este tipo de método sería:

```
<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>
```

En el ejemplo anterior, se usó un constructor `System.DateTime` (`new(2008, 4, 29)`) para crear un objeto .NET de tipo `System.DateTime`. Este objeto se creó dos veces, una vez como valor de la variable `releasedate`, y otra vez como primer y único argumento del método `System.DateTime.ToString()`. Al método de instancia `System.DateTime.ToString()` se le llama dos veces, ambas con el constructor `System.DateTime` (`new(2008, 4, 29)`) como primer y único argumento. En una de estas instancias, se usó la variable `releasedate` para obtener el objeto .NET.

Métodos de instancia y campos de instancia

La diferencia entre un método de instancia y un campo de instancia es solo teórica. En un método de instancia, se pasa directamente un objeto .NET como argumento. En un campo de instancia, se pasa un parámetro o una variable (aunque el parámetro o la variable puede contener un objeto .NET). Por ejemplo, en el

código del ejemplo anterior, la variable `releasedate` contiene un objeto .NET y esta es la variable que se pasa como argumento de `ToString()` en el segundo constructor de elemento `date`. Por tanto, la instancia `ToString()` del primer elemento `date` es un método de instancia, mientras que la segunda se considera un campo de instancia. El resultado es el mismo en ambos casos.

10.2.2.2.4 Tipos de datos: conversión de XPath/XQuery en .NET

Cuando se usa una función de extensión .NET dentro de una expresión XPath/XQuery, los tipos de datos de los argumentos de la función son importantes para determinar a cuál de los métodos .NET que tienen el mismo nombre se está llamando.

En .NET se siguen estas normas:

- Si en una clase hay varios métodos que tienen el mismo nombre, solamente se pueden seleccionar los métodos que tienen el mismo número de argumentos que la llamada a función.
- Los tipos de datos de cadena, numéricos y booleanos XPath/XQuery (*ver lista más abajo*) se convierten de forma implícita en el tipo de datos .NET correspondiente. Si el tipo XPath/XQuery suministrado se puede convertir en más de un tipo .NET (p. ej. `xs:integer`), entonces se selecciona el tipo .NET que se declaró para el método seleccionado. Por ejemplo, si el método .NET al que se está llamando es `fx(double)` y el tipo de datos XPath/XQuery suministrado es `xs:integer`, entonces se convierte `xs:integer` en el tipo de datos .NET `double`.

La tabla que aparece a continuación enumera las conversiones implícitas de los tipos de cadena, numéricos y booleanos XPath/XQuery en tipos de datos .NET.

`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`

Los subtipos de los tipos de datos XML Schema de la tabla anterior (que se usan en XPath y XQuery) también se convierten en los tipos .NET correspondientes al tipo antecesor del subtipo.

En algunos casos quizás no sea posible seleccionar el método .NET correcto usando la información dada. Por ejemplo, imagine que:

- El argumento presentado es un valor `xs:untypedAtomic` de 10 y está destinado al método `mimétodo(float)`.
- Sin embargo, hay otro método en la clase que toma un argumento de otro tipo de datos: `mimétodo(double)`.
- Puesto que los métodos tienen el mismo nombre y el tipo suministrado (`xs:untypedAtomic`) se puede convertir correctamente tanto en `float` como en `double`, es posible que `xs:untypedAtomic` se convierta en `double` en lugar de en `float`.

 © 2017-2023 Altova GmbH

- Por consiguiente, el método seleccionado no será el método necesario y puede que no produzca el resultado esperado. Una solución es crear un método definido por el usuario con un nombre diferente y usar ese método.

Los tipos que no aparecen en la lista anterior (p. ej. `xs:date`) no se convertirán y generarán un error.

10.2.2.2.5 Tipos de datos: conversión de .NET en XPath/XQuery

Cuando un método .NET devuelve un valor y el tipo de datos del valor es un tipo de cadena, numérico o booleano, entonces se convierte en el tipo de datos XPath/XQuery correspondiente. Por ejemplo, el tipo de datos .NET `decimal` se convierte en `xsd:decimal`.

Cuando se devuelve un objeto .NET o un tipo de datos que no es de cadena, numérico ni booleano, puede garantizar la conversión del tipo XPath/XQuery necesario usando primero un método .NET (p. ej. `System.DateTime.ToString()`) para convertir el objeto .NET en una cadena. En XPath/XQuery la cadena se puede modificar para ajustarse a la representación léxica del tipo necesario y convertirse después en dicho tipo (usando la expresión `cast as`, por ejemplo).

10.2.2.3 Scripts MSXSL para XSLT

El elemento `<msxsl:script>` contiene funciones y variables definidas por el usuario a las que se puede llamar desde dentro de expresiones XPath en la hoja de estilos XSLT. El elemento `<msxsl:script>` es un elemento de nivel superior, es decir, debe ser un elemento secundario de `<xsl:stylesheet>` o `<xsl:transform>`.

El elemento `<msxsl:script>` debe estar en el espacio de nombres `urn:schemas-microsoft-com:xslt` (*ver ejemplo más abajo*).

Lenguaje de scripting y espacio de nombres

El lenguaje de scripting utilizado dentro del bloque se especifica en el atributo `language` del elemento `<msxsl:script>` y el espacio de nombres que se debe usar para las llamadas a función desde expresiones XPath se identifica con el atributo `implements-prefix`:

```
<msxsl:script language="lenguaje-de-scripting" implements-prefix="prefijo-espacioNombres-
usuario">

   función-1 o variable-1
   ...
   función-n o variable-n

</msxsl:script>
```

El elemento `<msxsl:script>` interactúa con Windows Scripting Runtime, de modo que dentro del elemento `<msxsl:script>` solamente se pueden usar lenguajes que estén instalados en el equipo. Para poder usar

scripts MSXSL es necesario tener **instalada la plataforma .NET Framework 2.0 (o superior)**. Por tanto, los lenguajes de scripting .NET se pueden usar dentro del elemento `<msxsl:script>`.

El atributo `language` admite los mismos valores que el atributo `language` del elemento HTML `<script>`. Si no se especifica el atributo `language`, entonce se asume Microsoft JScript por defecto.

El atributo `implements-prefix` toma un valor que es un prefijo de un espacio de nombres declarado dentro del ámbito. Este espacio de nombres suele ser un espacio de nombres de usuario que se reservó para una biblioteca de funciones. Todas las funciones y variables definidas dentro del elemento `<msxsl:script>` están en el espacio de nombres identificado por el prefijo indicado en el atributo `implements-prefix`. Cuando se llama a una función desde dentro de una expresión XPath, el nombre de función completo debe estar en el mismo espacio de nombres que la definición de función.

Ejemplo

Aquí puede ver un ejemplo de una hoja de estilos XSLT que usa una función definida dentro de un elemento `<msxsl:script>`.

```
<?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>
```

 © 2017-2023 Altova GmbH

```
  </xsl:template>
</xsl:stylesheet>
```

Tipos de datos

Los valores de los parámetros que se pasan dentro y fuera del bloque de script solamente pueden ser tipos de datos XPath. Esta restricción no afecta a los datos que se pasan las funciones y variables situadas dentro del bloque de script.

Ensamblados

Puede importar un ensamblado al script usando el elemento `msxsl:assembly`. El ensamblado se identifica con un nombre o un URI. El ensamblado se importa cuando se compila la hoja de estilos. Aquí puede ver cómo se usa el elemento `msxsl:assembly`:

```
<msxsl:script>
       <msxsl:assembly name="miEnsamblado.nombreEnsamblado" />
       <msxsl:assembly href="rutaDelEnsamblado" />

       ...

</msxsl:script>
```

El nombre de ensamblado puede ser un nombre completo, como:

```
"system.Math, Version=3.1.4500.1 Culture=neutral PublicKeyToken=a46b3f648229c514"
```

o un nombre abreviado, como `"miEnsamblado.Draw"`.

Espacios de nombres

Puede declarar espacios de nombres con el elemento `msxsl:using`. Esto permite escribir las clases del ensamblado en el script sin sus espacios de nombres, lo cual le permitirá ahorrar mucho tiempo. Aquí puede ver cómo se usa el elemento `msxsl:using` para declarar espacios de nombres.

```
<msxsl:script>
       <msxsl:using namespace="ENmiEnsamblado.NombreEspaciodenombres" />

       ...

</msxsl:script>
```

El valor del atributo `namespace` es el nombre del espacio de nombres.

Índice

.

.NET Framework API, 381

A

Altova ServiceController, 30
API de Python, 371
API REST,
proyecto de ejemplo, 287
Archivo de configuración JSON,
para el módulo Python de RaptorXMLServer, 374
Assgning a license to RaptorXML Server on Linux, 38
Assgning a license to RaptorXML Server on macOS, 43
Assgning a license to RaptorXML Server on Windows, 32

B

Biblioteca Python,
de RaptorXML Server, 374

C

Catálogos, 48
Catálogos en RaptorXML, 50
Catálogos XML, 48
Catálogos y variables de entorno, 53
Clase contenedora para la interfaz REST, 288
Comando help de la ILC, 211
Comandos de licencias en la ILC, 217
Comandos XQuery, 96
Comandos XSLT, 128
Comprobar el formato, 83
Conexiones de red, 28
Configuración,
en Linux, 34
en macOS, 40
Configuración de servicios, 28
Configuración del servidor, 258
CoreCatalog.xml, 50
CustomCatalog.xml, 50

D

Depurar scripts de Python del lado servidor, 377
Depurar scripts de Python en Visual Studio Code, 378
Desinstalación, 23
Desinstalar, 23
Documentos XQuery,
validación, 114
Documentos XSLT,
validación, 137

E

Ejecución de XQuery, 96
Ejemplo en C# para API REST, 288
Esquemas,
buscar en los catálogos, 51
Esquemas DTD y catálogos, 48
Esquemas y catálogos, 48
Extensiones de Altova,
funciones para gráficos, 415

F

Funciones de extensión .NET,
campos de instancia, 525
campos estáticos, 524
constructores, 523
conversiones de tipos de datos, 526, 527
métodos de instancia, 525
métodos estáticos, 524
para XSLT y XQuery, 521
resumen, 521
tipos de datos .NET en XPath/XQuery, 527
tipos de datos XPath/XQuery en .NET, 526
Funciones de extensión .NET para XSLT y XQuery,
ver Funciones de extensión .NET, 521

© 2017-2023 Altova GmbH

Funciones de extensión en scripts MSXSL,
msxsl:script, 527
Funciones de extensión Java,
archivos de clases definidos por el usuario, 513
archivos JAR definidos por el usuario, 516
campos de instancia, 519
campos estáticos, 518
constructores, 517
conversiones de tipos de datos, 519, 520
métodos de instancia, 519
métodos estáticos, 518
para XSLT y XQuery, 512
resumen, 512
tipos de datos Java en XPath/XQuery, 520
tipos de datos XPath/XQuery en Java, 519
Funciones de extensión Java para XSLT y XQuery,
ver Funciones de extensión Java, 512
Funciones de extensión para XSLT y XQuery, 511
Funciones para gráficos,
ejemplo, 505
estructura de datos de gráficos, 500
fragmento de código, 495

G

Gestor de esquemas,
aplicar un parche a un esquema, 391
desinstalar un esquema, 393
estado de los esquemas en el, 389
funcionamiento del, 386
help (comando ILC), 394
info (comando ILC), 395
initialize (comando ILC), 395
instalar un esquema, 391
install (comando ILC), 396
introducción, 382
introducción a la ILC, 394
list (comando ILC), 396
lista de esquemas por estado, 389
reset (comando ILC), 397
restaurar, 393
uninstall (comando ILC), 398
update (comando ILC), 399
upgrade (comando ILC), 399

I

Iniciar LicenseServer en Linux, 37
Iniciar LicenseServer en macOS, 42
Iniciar LicenseServer en Windows, 30
Iniciar RaptorXML Server en Linux, 37
Iniciar RaptorXML Server en macOS, 42
Iniciar RaptorXML Server en Windows, 30
Instalación, 47
Windows, 23
Instalación en macOS, 40
Instalación en Windows, 23
Instalación en Windows Server Core, 24
Propiedades del servicio, 27
propiedades del servidor web, 26
propiedades del servidor web SSL, 26
Instalar el módulo Python de RaptorXMLServer, 374
Instalar LicenseServer en Linux, 36
Instalar LicenseServer en macOS, 41
Instalar LicenseServer en Windows, 27
Installation sur Linux, 34
Interfaces,
resumen, 11
Interfaz .NET, 11
Interfaz COM, 11
Interfaz HTTP, 11, 254
configuración del servidor, 258
preparar el servidor, 255
problemas de seguridad, 57
proyecto de ejemplo, 287
solicitudes cliente, 267
Interfaz Java, 11
Interfaz Python, 11
Interfaz REST,
clases contenedoras, 288

L

License for RaptorXML Server,
assigning on Linux, 38
assigning on macOS, 43
assigning on Windows, 32
Licensing RaptorXML Server on Linux, 37

© 2017-2023 Altova GmbH

Licensing RaptorXML Server on macOS, 42
Licensing RaptorXML Server on Windows, 29
Línea de comandos,
opciones, 236
resumen de uso, 58
y XQuery, 96
Linux,
installation sur, 34
licensing RaptorXML Server on, 37
Localización, 213

M

macOS,
Instalación en, 40
licensing RaptorXML Server on, 42
upgrading RaptorXML Server on, 45
Migrar RaptorXML Server a un equipo nuevo, 46
Módulo Python,
de RaptorXML Server, 374

P

Personalizar Catálogos, 51
pip (comando), 374
Preguntas frecuentes sobre la API de Python, 379
Problemas de seguridad, 57
Python,
problemas de seguridad, 57

R

RaptorXML,
características, 17
ediciones e interfaces, 11
especificaciones compatibles, 19
interfaces con COM, Java y .NET, 11
interfaz de la línea de comandos, 11
interfaz HTTP, 11
interfaz Python, 11
introducción, 10
requisitos del sistema, 15
RaptorXML Server,
migrar a un equipo nuevo, 46
Recursos globales, 55
Registrar RaptorXML Server con LicenseServer en Linux, 38
Registrar RaptorXML Server con LicenseServer en macOS, 43
Registrar RaptorXML Server con LicenseServer en Windows, 32
Resumen del mecanismo de catalogación, 48
RootCatalog.xml, 50, 374

S

Scripts de Python en RaptorXML Server, 377
Scripts en XSLT/XQuery,
ver Funciones de extensión, 511

T

Transformación XSLT, 128

U

Upgrading RaptorXML Server on macOS, 45

V

Validación,
de cualquier documento, 209
de documentos XQuery, 114
de documentos XSLT, 137
de DTD, 71
de instancias XML con DTD, 59
de instancias XML con XSD, 63
de XSD, 75
Variables de entorno, 53
Variables de entorno usadas en los catálogos, 50
Versiones de LicenseServer, 27, 36, 41
Visual Studio y scripts de Python, 378

© 2017-2023 Altova GmbH

W

Windows,
Instalación en, 23
licensing RaptorXML Server on, 29

X

XQuery,
funciones de extensión, 511
XSLT,
funciones de extensión, 511

© 2017-2023 Altova GmbH