Manual de referencia y del usuario

Manual del usuario y referencia de Altova RaptorXML 2013

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: 2013

© 2013 Altova GmbH

Tabla de contenido

1 Introducción a RaptorXML 3

1.1 Ediciones e interfaces ... 5
1.2 Requisitos del sistema ... 8
1.3 Características ... 9
1.4 Especificaciones compatibles ... 11

2 Configurar RaptorXML 14

2.1 En Windows ... 15
2.2 En Linux ... 16
2.2.1 Instalación en Linux ... 17
2.2.2 Asignar licencias a RaptorXML en Linux ... 19
2.3 En Mac OS X ... 21
2.3.1 Instalación en Mac OS X ... 22
2.3.2 Asignar licencias a RaptorXML en Mac OS X ... 24
2.4 Catálogos XML ... 26
2.4.1 Cómo funcionan los catálogos ... 27
2.4.2 Mecanismo de catalogación XML de Altova ... 29
2.4.3 Variables para ubicaciones de sistemas Windows ... 32
2.5 Recursos globales ... 33
2.6 Problemas de seguridad ... 35

3 Interfaz de la línea de comandos (ILC) 38

3.1 Comandos para validar XML, DTD, XSD ... 40
3.1.1 valxml-withdtd (xml) ... 41
3.1.2 valxml-withxsd (xsi) ... 43
3.1.3 valdtd (dtd) ... 45
3.1.4 valxsd (xsd) ... 47
3.1.5 valany ... 49
3.2 Comandos para comprobar el formato ... 51
3.2.1 wfxml ... 52
3.2.2 wfdtd ... 54
3.2.3 wfany ... 56

3.3 Comandos XSLT ... 58
3.3.1 xslt ... 59
3.3.2 valxslt ... 61
3.4 Comandos XQuery ... 63
3.4.1 xquery ... 64
3.4.2 valxquery ... 66
3.5 Comandos de ayuda y licencias ... 68
3.5.1 Ayuda ... 69
3.5.2 Licencias ... 70
3.6 Comandos de localización ... 71
3.6.1 exportresourcestrings ... 72
3.6.2 setdeflang ... 73
3.7 Opciones ... 74
3.7.1 Catálogos ... 75
3.7.2 Errores ... 76
3.7.3 Recursos globales ... 77
3.7.4 Ayuda y versión ... 78
3.7.5 Mensajes ... 79
3.7.6 Procesamiento ... 80
3.7.7 Instancias XML ... 81
3.7.8 Validación de instancias XML ... 82
3.7.9 Esquemas XML (XSD) ... 83
3.7.10 XQuery ... 86
3.7.11 XSLT ... 88
3.7.12 Archivos ZIP ... 90

4 Interfaz HTTP 92

4.1 Preparar el servidor ... 94
4.1.1 Iniciar el servidor ... 95
4.1.2 Probar la conexión ... 97
4.1.3 Configurar el servidor ... 98
4.2 Solicitudes cliente ... 102
4.2.1 Iniciar trabajos con POST ... 105
4.2.2 Respuesta del servidor a solicitudes POST ... 109
4.2.3 Obtener el documento de resultados ... 112
4.2.4 Obtener los documentos de salida/errores ... 116
4.2.5 Liberar espacio tras el procesamiento ... 118

5 Interfaz Python 120

5.1 Crear scripts Python ... 121
5.2 Ejecutar scripts Python ... 124
5.3 Ejemplo de script 1: procesar XML ... 125
5.3.1 Script ... 126
5.3.2 Documento de resultados ... 128
5.4 Ejemplo de script 2: cambiar el formato XML ... 129
5.4.1 Script ... 130
5.4.2 Documento de resultados ... 133
5.5 API de Python: el objeto Job ... 136
5.6 API de Python para XML ... 137
5.6.1 xml.Document ... 139
5.6.2 xml.Element ... 140
5.6.3 xml.Attribute ... 142
5.6.4 xml.NSAttribute ... 143
5.6.5 xml.ProcessingInstruction ... 144
5.6.6 xml.UnexpandedEntityReference ... 145
5.6.7 xml.Character ... 146
5.6.8 xml.Comment ... 147
5.6.9 xml.UnparsedEntity ... 148
5.6.10 xml.Notation ... 149
5.6.11 xml.Namespace ... 150
5.6.12 xml.QName ... 151
5.7 API de Python para XSD ... 152
5.7.1 xsd.Annotation ... 157
5.7.2 xsd.Any ... 158
5.7.3 xsd.AnyAttribute ... 159
5.7.4 xsd.Assertion ... 160
5.7.5 xsd.AttributeDeclaration ... 161
5.7.6 xsd.AttributeGroupDefinition ... 162
5.7.7 xsd.AttributePSVI ... 163
5.7.8 xsd.AttributeUse ... 165
5.7.9 xsd.Block ... 166
5.7.10 xsd.ComplexTypeDefnition ... 167
5.7.11 xsd.ContentType ... 168
5.7.12 xsd.Defined ... 169
5.7.13 xsd.DerivationMethod ... 170
5.7.14 xsd.ENTITY ... 171

5.7.15 xsd.ElementDeclaration ... 172
5.7.16 xsd.ElementPSVI ... 173
5.7.17 xsd.Final ... 175
5.7.18 xsd.ID ... 176
5.7.19 xsd.IDREF ... 177
5.7.20 xsd.ID_IDREF_binding ... 178
5.7.21 xsd.ID_IDREF_table ... 179
5.7.22 xsd.IdentityConstraintDefinition ... 180
5.7.23 xsd.Instance ... 181
5.7.24 xsd.ModelGroup ... 182
5.7.25 xsd.ModelGroupDefinition ... 183
5.7.26 xsd.NCName ... 184
5.7.27 xsd.NMTOKEN ... 185
5.7.28 xsd.NOTATION ... 186
5.7.29 xsd.Name ... 187
5.7.30 xsd.NamespaceBinding ... 188
5.7.31 xsd.NamespaceConstraint ... 189
5.7.32 xsd.NotationDeclaration ... 190
5.7.33 xsd.OpenContent ... 191
5.7.34 xsd.PSVI ... 192
5.7.35 xsd.Particle ... 193
5.7.36 xsd.QName ... 194
5.7.37 xsd.Schema ... 195
5.7.38 xsd.Scope ... 196
5.7.39 xsd.Sibling ... 197
5.7.40 xsd.SimpleTypeDefinition ... 198
5.7.41 xsd.TypeAlternative ... 200
5.7.42 xsd.TypeTable ... 201
5.7.43 xsd.Unbounded ... 202
5.7.44 xsd.ValueConstraint ... 203
5.7.45 xsd.XPathExpression ... 204
5.7.46 Objetos de tipos de datos especiales integrados ... 205
5.7.47 Objetos de tipo de datos string ... 206
5.7.48 Objeto de tipo de datos boolean ... 207
5.7.49 Objetos de tipo de datos number ... 208
5.7.50 Objetos de tipo de datos duration ... 209
5.7.51 Objetos de tipo de datos date y time ... 210
5.7.52 Objetos de tipo de datos binary ... 211
5.7.53 Objetos faceta ... 212

6 Interfaz Java 216

6.1 RaptorXMLJava - RaptorXMLFactory ... 217
6.2 RaptorXMLJava - XBRL ... 222
6.3 RaptorXMLJava - XMLValidator ... 230
6.4 RaptorXMLJava - XQuery ... 244
6.5 RaptorXMLJava - XSLT ... 252
6.6 RaptorXMLJava - RaptorXMLException ... 261

7 Interfaz .NET/COM 264

7.1 RaptorXMLDev_COM - ENUMAssessmentMode ... 266
7.2 RaptorXMLDev_COM - ENUMErrorFormat ... 267
7.3 RaptorXMLDev_COM - ENUMLoadSchemalocation ... 268
7.4 RaptorXMLDev_COM - ENUMQueryVersion ... 270
7.5 RaptorXMLDev_COM - ENUMSchemaImports ... 271
7.6 RaptorXMLDev_COM - ENUMSchemaMapping ... 273
7.7 RaptorXMLDev_COM - ENUMValidationType ... 274
7.8 RaptorXMLDev_COM - ENUMWellformedCheckType ... 275
7.9 RaptorXMLDev_COM - ENUMXBRLValidationType ... 276
7.10 RaptorXMLDev_COM - ENUMXMLValidationMode ... 277
7.11 RaptorXMLDev_COM - ENUMXQueryVersion ... 278
7.12 RaptorXMLDev_COM - ENUMXSDVersion ... 279
7.13 RaptorXMLDev_COM - ENUMXSLTVersion ... 280
7.14 RaptorXMLDev_COM - IApplication ... 281
7.15 RaptorXMLDev_COM - IXBRL ... 285
7.16 RaptorXMLDev_COM - IXMLValidator ... 292
7.17 RaptorXMLDev_COM - IXQuery ... 299
7.18 RaptorXMLDev_COM - IXSLT ... 306

8 Información sobre motores XSLT y XQuery 316

8.1 XSLT 1.0 ... 317
8.2 XSLT 2.0 ... 318
8.3 XSLT 3.0 ... 321
8.4 XQuery 1.0 ... 322
8.5 XQuery 3.0 ... 326

8.6 Funciones XQuery 1.0 y XPath 2.0 ... 327
8.7 Funciones XQuery y XPath 3.0 ... 331

9 Funciones de extensión XSLT y XQuery 334

9.1 Funciones de extensión de Altova ... 335
9.1.1 Funciones generales ... 336
9.1.2 Funciones para códigos de barras ... 340
9.1.3 Funciones para gráficos ... 342
9.1.4 Estructura XML de los datos de gráfico ... 346
9.1.5 Ejemplo: funciones de gráficos ... 351
9.2 Funciones de extensión Java ... 354
9.2.1 Archivos de clases definidos por el usuario ... 356
9.2.2 Archivos JAR definidos por el usuario ... 359
9.2.3 Java: constructores ... 360
9.2.4 Java: métodos estáticos y campos estáticos ... 361
9.2.5 Java: métodos de instancia y campos de instancia ... 362
9.2.6 Tipos de datos: XPath/XQuery a Java ... 363
9.2.7 Tipos de datos: Java a XPath/XQuery ... 364
9.3 Funciones de extensión .NET ... 365
9.3.1 .NET: constructores ... 368
9.3.2 .NET: métodos estáticos y campos estáticos ... 369
9.3.3 .NET: métodos de instancia y campos de instancia ... 370
9.3.4 Tipos de datos: XPath/XQuery a .NET ... 371
9.3.5 Tipos de datos: .NET a XPath/XQuery ... 372
9.4 Scripts MSXSL para XSLT ... 373

10 Altova LicenseServer 378

10.1 Información de red ... 379
10.2 Instalación en Windows ... 380
10.3 Instalación en Linux ... 381
10.4 Instalación en Mac OS X ... 382
10.5 Altova ServiceController ... 383
10.6 Asignación de licencias ... 384
10.6.1 Iniciar LicenseServer ... 385
10.6.2 Abrir la página de configuración de LicenseServer (Windows) ... 387
10.6.3 Abrir la página de configuración de LicenseServer (Linux) ... 390
10.6.4 Abrir la página de configuración de LicenseServer (Mac OS X) ... 392
10.6.5 Cargar licencias en LicenseServer ... 394

10.6.6 Registrar FlowForce Server con LicenseServer 397
10.6.7 Registrar MapForce Server con LicenseServer 401
10.6.8 Registrar RaptorXML(+XBRL) Server con LicenseServer 403
10.6.9 Registrar StyleVision Server con LicenseServer 405
10.6.10 Asignar licencias a productos registrados 407
10.7 Referencia de la página de configuración 412
10.7.1 Repertorio de licencias 413
10.7.2 Gestión de servidores 417
10.7.3 Supervisión de servidores 420
10.7.4 Opciones de configuración 421
10.7.5 Mensajes, Cerrar sesión 424

Altova RaptorXML 2013

Introducción a RaptorXML

1 Introducción a RaptorXML

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 ni en RaptorXML Development Edition).

Ediciones y sistemas operativos
Altova ofrece tres ediciones diferentes de RaptorXML, diseñadas para satisfacer diferentes requisitos. Las tres ediciones de RaptorXML se describen en el apartado Ediciones e interfaces . Mientras que la edición Development Edition solamente es compatible con el sistema operativo Windows, las dos ediciones servidor están disponibles para Windows, Linux y Mac OS X. Para más información consulte el apartado Requisitos del sistema.

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. En el apartado Especificaciones compatibles se enumeran todas las especificaciones con las que cumple RaptorXML. Para más información visite el sitio web de Altova.

Esta documentación
La presente documentación está incluida en la aplicación y también está disponible en el sitio web de Altova. Tenga en cuenta que el explorador Chrome tiene una restricción que no permite expandir las entradas de la tabla de contenido cuando la documentación se abre localmente. Sin embargo, si abre la documentación desde un servidor web, la tabla de contenido funciona correctamente en Chrome.

La presente documentación se divide en varias secciones:

- Introducción a RaptorXML (la presente sección)
- Configurar RaptorXML
- Interfaz de la línea de comandos
- Interfaz HTTP
- Interfaz Python
- Interfaz Java
- Interfaz COM/.NET
- Información sobre motores XSLT y XQuery
- Funciones de extensión XSLT y XQuery
- Altova LicenseServer

Última actualización: 23/07/2013

© 2013 Altova GmbH

1.1 Ediciones e interfaces

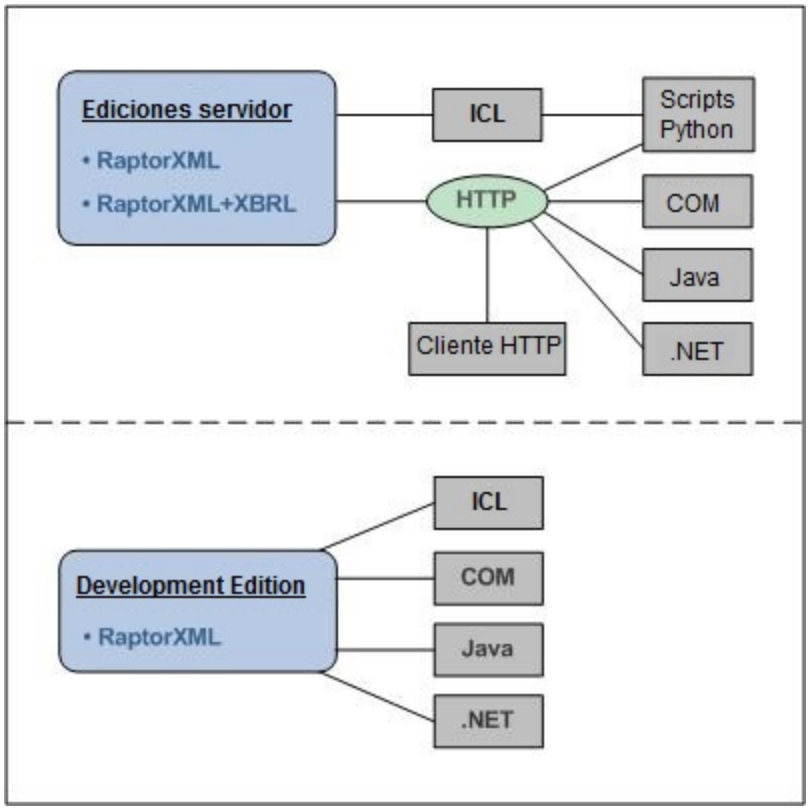
Altova ofrece tres ediciones distintas de RaptorXML:

- ***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.
- ***RaptorXML Development Edition:*** disponible para sistemas Windows. Se puede descargar por separado desde el sitio web de Altova y se puede activar con la licencia de los productos de Altova MissionKit (XMLSpy, MapForce y StyleVision). Al igual que RaptorXML(+XBRL) Server, puede utilizarse para validaciones y transformaciones XML, XSLT y XQuery, pero sus funciones son limitadas en comparación con RaptorXML(+XBRL) Server. RaptorXML Development Edition:
  - Solamente se puede ejecutar una instancia de su binario al mismo tiempo en el mismo equipo.
  - Solamente es compatible con equipos Windows (no es compatible con Windows Server, Linux ni Mac OS X).
  - No es compatible con multi-threading ni procesamiento por lotes (procesamiento de múltiples archivos)
  - No es compatible con la representación de gráficos
  - No es compatible con XBRL
  - Solamente está disponible en una versión de 32 bits

Interfaces
Puede acceder a RaptorXML a través de varias interfaces:

- Una interfaz de la línea de comandos (en adelante ***ILC***) *(en todas las ediciones)*
- Una **interfaz COM** para sistemas Windows *(en todas las ediciones)*
- Una **interfaz .NET** para sistemas Windows *(en todas las ediciones)*
- Una **interfaz Java** para sistemas Windows, Linux y Mac OS *(en todas las ediciones)*
- Una **interfaz HTTP** a la que se puede acceder desde un cliente HTTP *(solo en las ediciones servidor)*
- Una **interfaz Python** con la que puede acceder y procesar partes de documentos mediante scripts Python y con ayuda de las API de Python de RaptorXML. Los scripts se pueden enviar por la interfaz de la línea de comandos o por la interfaz HTTP (solo en las ediciones servidor).

El diagrama que aparece a continuación muestra cómo se accede a las dos ediciones servidor (RaptorXML Server y RaptorXML+XBRL Server) y a la edición limitada RaptorXML Development Edition a través de las diferentes interfaces.



Observe que las interfaces COM, Java y .NET usan el protocolo HTTP para conectarse a las ediciones servidor. Los scripts Pyton se pueden enviar a las ediciones servidor a través de la interfaz HTTP y de la línea de comandos.

Interfaz de la línea de comandos (ILC)
Permite validar XML (y otros documentos), transformar XSLT y ejecutar XQuery desde la línea de comandos. Para más información y aprender a usarla consulte la sección Interfaz de la línea de comandos.

Interfaz HTTP
Puede acceder a todas las funciones de las ediciones servidor a través de una interfaz HTTP. Las solicitudes cliente se hacen en formato JSON. Cada solicitud se asigna a un directorio de trabajo en el servidor (en este directorio se guardan los archivos de salida) y las respuestas del servidor al cliente incluyen toda la información relacionada con el trabajo. Para más información consulte la sección Interfaz HTTP.

Interfaz Python
Junto con un comando de la ILC o una solicitud HTTP, puede suministrar un script Python que acceda a documentos especificados en el comando o en la solicitud. El acceso al documento se consigue a través de las API Python para XML, XSD y XBRL. Para más información sobre esta interfaz y aprender a usarla consulte la sección Interfaz Python.

Interfaz COM
Puede usar RaptorXML a través de la interfaz COM y, por tanto, puede ser utilizada por aplicaciones y lenguajes de script compatibles con COM. La compatibilidad con al interfaz COM

 © 2013 Altova GmbH

está implementada para interfaces sin formato e interfaces de envío. Los datos de entrada se pueden suministrar como archivos o como cadenas de texto en scripts y en los datos de la aplicación.

Interfaz Java
Las funciones de RaptorXML también están disponibles como clases Java que se pueden usar en programas Java. Por ejemplo, hay clases Java que ofrecen características de validación XML, transformación XSLT y ejecución de XQuery.

Interfaz .NET
RaptorXML ofrece un archivo DLL construido como contenedor de RaptorXML que permite a los usuarios de .NET conectarse a las funciones de RaptorXML. Además, RaptorXML ofrece un ensamblado de interoperabilidad principal firmado por Altova. Los datos de entrada se pueden suministrar como archivos o como cadenas de texto en scripts y en los datos de la aplicación.

1.2 Requisitos del sistema

RaptorXML es compatible con estos sistemas operativos:

- Windows Server 2008 R2 o superior
- Windows XP con Service Pack 3, Windows 7, Windows 8 o superior
- Linux (CentOS 6, RedHat 6, Debian 6 y Ubuntu 12.04 o superior)
- Mac OS X 10.7 o superior

RaptorXML(+XBRL) Server 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. RaptorXML Development Edition solamente está disponible en una versión de 32 bits.

Para usar RaptorXML 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.

© 2013 Altova GmbH

1.3 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 XBRL, 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.

Características de alto rendimiento

- Optimizaciones de código de altísimo rendimiento
  - Implementaciones nativas de conjuntos de instrucciones
  - Versión de 32 y 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
- 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 y visite el sitio web de Altova.

© 2013 Altova GmbH

1.4 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)
- XML Path Language (XPath) 3.0

Borradores y recomendaciones candidatas del W3C
Sitio web: World Wide Web Consortium (W3C)

- XSL Transformations (XSLT) Version 3.0
- XQuery 3.0: An XML Query Language
- XPath and XQuery Functions and Operators 3.0

Estándares OASIS
Sitio web: OASIS Standards

- XML Catalogs V 1.1 - OASIS Standard V1.1

Altova RaptorXML 2013

Configurar RaptorXML

2 Configurar RaptorXML

Esta sección describe el procedimiento para instalar y configurar RaptorXML Server correctamente. Incluye varios apartados con información sobre:

- Cómo instalar RaptorXML y asignarle licencias en sistemas Windows, Linux y on Mac OS X.
- Cómo usar los catálogos XML.
- Cómo trabajar con los recursos globales de Altova.
- Problemas de seguridad relacionados con RaptorXML.

RaptorXML tiene opciones especiales que admiten el uso de catálogos XML y de recursos globales de Altova, características que mejoran la portabilidad y modularidad del entorno en el que se trabaja.

Nota: el apartado Problemas de seguridad explica cómo configurar importantes soluciones de seguridad.

© 2013 Altova GmbH

2.1 En Windows

Temas de este apartado:

- Instalación en Windows
- Asignar licencias en Windows

Instalación en Windows
RaptorXML se puede descargar del centro de descargas del sitio web de Altova. El paquete de instalación instala RaptorXML con los registros necesarios en esta carpeta:

```
<CarpetaArchivosPrograma>\Altova\RaptorXMLServer2013\bin\RaptorXML.exe
```

Todos los registros necesarios para usar RaptorXML a través de una interfaz COM, como interfaz Java y en el entorno .NET los hace el programa de instalación, lo cual incluye registrar el ejecutable de RaptorXML como objeto de servidor COM, instalar `RaptorXMLLib.dll` (para el uso como interfaz Java) en el directorio `WINDIR\system32\` y añadir el archivo `Altova.RaptorXML.dll` a la biblioteca de referencia de .NET.

Asignar licencias en Windows
RaptorXML necesita recibir una licencia desde un servidor de licencias Altova LicenseServer, instalado en su equipo o en su red (consulte la documentación de LicenseServer). Tras instalar LicenseServer, ejecute el comando `licenseserver` de la interfaz de la línea de comandos para registrar RaptorXMLServer con Altova LicenseServer. El comando toma como argumento el nombre o la dirección IP del servidor en el que se ejecuta LicenseServer.

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

Tras registrar RaptorXML con LicenseServer, se devuelve la URL de la interfaz web de LicenseServer. Escriba la URL en una ventana del explorador para acceder a la interfaz web y después siga los pasos descritos en la documentación de LicenseServer para asignar una licencia a RaptorXML.

2.2 En Linux

Puede instalar RaptorXML Server en sistemas Linux (CentOS 6, RedHat 6, Debian 6 y Ubuntu 12.04 o superior). Este apartado describe el proceso de instalación y cómo asignarle licencias desde LicenseServer:

- Instalación en Linux
- Asignar licencias en RaptorXML en Linux

© 2013 Altova GmbH

2.2.1 Instalación en Linux

Temas de este apartado:

- Desinstalar versiones previas de RaptorXML y LicenseServer
- Copiar el paquete Linux en un directorio Linux
- Instalar RaptorXML
- Instalar LicenseServer

En este apartado describimos primero el proceso de desinstalación, por si necesita desinstalar una versión previa del software.

Desinstalar versiones previas de RaptorXML Server y LicenseServer
En la interfaz de la línea de comandos de Linux puede comprobar si ya están instalados RaptorXML Server y LicenseServer:

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

Si RaptorXML Server no está instalado todavía, siga las instrucciones que aparecen más abajo para instalarlo. Si RaptorXML Server ya está instalado y desea instalar una versión más reciente del software, primero debe desinstalar la versión previa con este comando:

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

Si necesita desinstalar una versión previa de Altova LicenseServer, utilice este comando:

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

Copiar el paquete de Linux
Descargue el paquete de instalación para Linux del sitio web de Altova y cópielo en cualquier directorio del sistema Linux. Para poder usar RaptorXML Server, necesitará Altova LicenseServer, que también se puede descargar del sitio web de Altova.

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

Instalación de RaptorXML Server

En la ventana de la Terminal cambie al directorio donde copió el paquete de Linux. Por ejemplo, si lo copió a un directorio llamado `MiAltova` (en `/home/User`, por ejemplo), cambie a ese directorio con este comando:

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

Instale RaptorXML Server con este comando:

```
[Debian]:   sudo dpkg --install raptorxmlserver-2013-debian.deb
[Ubuntu]:   sudo dpkg --install raptorxmlserver-2013-ubuntu.deb
[CentOS, RedHat]:   sudo rpm -ivh raptorxmlserver-2013-1.x86_64.rpm
```

El paquete de RaptorXML Server se instalará en esta carpeta:

```
/opt/Altova/raptorxmlserver2013
```

Instalación de Altova LicenseServer

Para poder ejecutar RaptorXML Server es necesario asignarle una licencia con un servidor Altova LicenseServer de la red. Descargue Altova LicenseServer del sitio web de Altova y cópielo en un directorio del sistema Linux y utilice este comando:

```
[Debian]:   sudo dpkg --install licenseserver-1.1-debian.deb
[Ubuntu]:   sudo dpkg --install licenseserver-1.1-ubuntu.deb
[CentOS, RedHat]:   sudo rpm -ivh licenseserver-1.1-1.x86_64.rpm
```

El paquete de LicenseServer se instalará en esta carpeta:

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

Para más información consulte el apartado siguiente Asignar licencias a RaptorXML.

 © 2013 Altova GmbH

2.2.2 Asignar licencias a RaptorXML en Linux

Temas de este apartado:

- Proceso de asignación de licencias
- Iniciar LicenseServer como servicio
- Registrar RaptorXML con LicenseServer
- Nota sobre núcleos y licencias

Proceso de asignación de licencias
Para asignar licencias a RaptorXML Server en sistemas Linux (Debian, Ubuntu, CentOS, RedHat):

1. Ejecute LicenseServer como servicio si todavía no está en ejecución.
2. Registre RaptorXML Server con LicenseServer.
3. En la página de configuración de LicenseServer, asigne una licencia a RaptorXML Server en función del número de núcleos del equipo que está ejecutando RaptorXML Server. Este proceso se describe en la documentación de Altova LicenseServer.

Nota: para usar RaptorXML Server es necesario tener un servidor de licencias Altova LicenseServer. Para más información consulte el apartado Instalación en Linux.

Nota: para registrar RaptorXML Server con LicenseServer es necesario tener privilegios de administrador (root).

Iniciar LicenseServer como servicio
Para registrar y asignar una licencia correctamente a RaptorXML Server, LicenseServer debe estar en ejecución como servicio. Inicie LicenseServer como servicio con este comando:

```
[Debian]                  sudo /etc/init.d/licenseserver start
[Ubuntu, CentOS, RedHat]  sudo initctl start licenseserver
```

Si por cualquier motivo necesita detener LicenseServer, use este comando:

```
[Debian]                  sudo /etc/init.d/licenseserver stop
[Ubuntu, CentOS, RedHat]  sudo initctl stop licenseserver
```

Registrar RaptorXML Server
Antes de poder asignarle una licencia, RaptorXML Server debe estar registrado con LicenseServer. Puede usar el comando **licenseserver** para registrar RaptorXML Server. Recuerde que debe iniciar RaptorXML Server con privilegios de root.

```
sudo /opt/Altova/RaptorXMLServer2013/bin/RaptorXML licenseserver localhost
```

En el comando anterior **localhost** es el nombre del servidor en el que está instalado

LicenseServer. Observe además que el ejecutable de RaptorXML Server está aquí:

```
/opt/Altova/RaptorXMLServer2013/bin
```

Tras registrar RaptorXML Server, puede ir a LicenseServer y asignarle una licencia. Esto se explica en la documentación de Altova LicenseServer.

Nota sobre núcleos y licencias
La asignación de licencias para RaptorXML Server depende del número de núcleos disponibles en el equipo donde se ejecuta RaptorXML Server. El número de núcleos que tienen una licencia asignada debe ser mayor o igual al número de núcleos disponibles en el servidor, ya se trate de un equipo virtual o físico. Por ejemplo, si su equipo servidor tiene ocho núcleos, debe comprar una licencia para ocho núcleos como mínimo.

Si usa un equipo servidor con gran cantidad de núcleos CPU, 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 será menos rápida que si se utilizaran todos los núcleos disponibles en el servidor.

© 2013 Altova GmbH

2.3 En Mac OS X

Puede instalar RaptorXML Server en sistemas Mac OS X (versión 10.7 o superior). En este apartado describimos el proceso de instalación y cómo asignarle licencias en sistemas Mac OS X:

- Instalación en Mac OS X
- Asignar licencias a RaptorXML en Mac OS X

2.3.1 Instalación en Mac OS X

Temas de este apartado:

- Desinstalar versiones previas de RaptorXML y LicenseServer
- Descargar el paquete de Mac OS X en un directorio de Mac OS X
- Instalar RaptorXML
- Instalar LicenseServer

En este apartado describimos primero el proceso de desinstalación, por si necesita desinstalar una versión previa del software.

Desinstalar versiones previas de RaptorXML Server y LicenseServer
Antes de desinstalar RaptorXML Server debe detener el servicio con este comando:

```
sudo launchctl unload /Library/LaunchDaemons/com.altova.RaptorXMLServer2013
.plist
```

Para comprobar si el servicio se detuvo, abra la terminal del Monitor de actividad y confirme que RaptorXML Server no está en la lista.

En la terminal de Aplicaciones haga clic con el botón secundario 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 puede utilizar este comando:

```
sudo rm -rf /usr/local/Altova/RaptorXMLServer2013/
```

Si necesita desinstalar una versión previa de Altova LicenseServer, utilice el procedimiento descrito para RaptorXML Server y este comando:

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

Descargar el paquete de instalación para Mac OS X
Descargue el paquete de instalación para Mac OS X del sitio web de Altova y cópielo en cualquier directorio. Para poder ejecutar RaptorXML Server es necesario tener instalado Altova LicenseServer, que también se puede descargar del sitio web de Altova. El archivo de instalación para Mac OS X tiene la extensión `.pkg`.

Instalación de RaptorXML Server
Abra el directorio donde copió el paquete de instalación y haga doble clic en el instalador. Siga los pasos del asistente para la instalación y acepte el contrato de licencia.

RaptorXML Server se instala en esta carpeta:

```
/usr/local/Altova/RaptorXMLServer2013/
```


© 2013 Altova GmbH

Haga clic en el icono de RaptorXML Server de la terminal de Aplicaciones para ver la ayuda en pantalla.

Instalación de Altova LicenseServer
Para poder ejecutar RaptorXML Server es necesario asignarle una licencia desde un servidor Altova LicenseServer de la red. Descargue LicenseServer del sitio web de Altova y haga doble clic en el instalador para iniciar la instalación. Siga los pasos del asistente para la instalación y acepte el contrato de licencia.

Altova LicenseServer se instala en esta carpeta:

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

Para más información sobre cómo registrar RaptorXML Server con Altova LicenseServer y asignarle licencias, consulte el apartado Asignar licencias a RaptorXML en Mac OS X.

2.3.2 Asignar licencias a RaptorXML en Mac OS X

Temas de este apartado:

- Proceso de asignación de licencias
- Iniciar LicenseServer como servicio
- Iniciar RaptorXML como servicio
- Registrar RaptorXML con LicenseServer
- Nota sobre núcleos y licencias

Proceso de asignación de licencias
Siga estos pasos para asignar licencias a RaptorXML Server desde sistemas Mac OS X:

1. Si LicenseServer no está en ejecución todavía, inícielo como servicio.
2. Inicie RaptorXML Server como servicio.
3. Registre RaptorXML Server con LicenseServer.
4. En la página de configuración de LicenseServer, asigne una licencia al equipo en que se está ejecutando RaptorXML Server. Esto se explica en la documentación de Altova LicenseServer.

Nota: es necesario tener %RXML%> Server y Altova LicenseServer instalados y ejecutándose como servicios. Para más información consulte el apartado Instalación en Mac OS X.

Nota: para poder registrar %RXML%> Server con LicenseServer necesita tener privilegios de administrador (root).

Iniciar LicenseServer como servicio
Para registrar y asignar una licencia a RaptorXML Server, LicenseServer debe estar en ejecución como servicio. Inicie LicenseServer como servicio con este comando:

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

Si por cualquier motivo necesita detener LicenseServer, use este comando:

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

Iniciar RaptorXML Server como servicio
Inicie RaptorXML Server como servicio con este comando:

```
sudo launchctl load /Library/LaunchDaemons/com.altova.RaptorXMLServer2013
.plist
```

Si por cualquier motivo necesita detener RaptorXML Server, use este comando:

```
sudo launchctl unload /Library/LaunchDaemons/com.altova.RaptorXMLServer2013
```


© 2013 Altova GmbH

```
.plist
```

Registrar RaptorXML Server
Para poder asignarle una licencia, antes debe registrar RaptorXML Server con LicenseServer. Para ello puede utilizar el comando `licenseserver` de la interfaz de la línea de comandos. Recuerde que RaptorXML Server debe iniciarse con privilegios de administrador (root).

```
sudo /usr/local/Altova/RaptorXMLServer2013/bin/RaptorXML licenseserver
localhost
```

En el comando anterior **`localhost`** es el nombre del servidor donde está instalado LicenseServer. Observe también que la ubicación del ejecutable de RaptorXML Server es esta:

```
/usr/local/Altova/RaptorXMLServer2013/bin
```

Tras registrar RaptorXML Server, puede ir a LicenseServer y asignarle una licencia. Esto se explica en la documentación de Altova LicenseServer.

Nota sobre núcleos y licencias
La asignación de licencias para RaptorXML Server depende del número de núcleos disponibles en el equipo donde se ejecuta RaptorXML Server. El número de núcleos que tienen una licencia asignada debe ser mayor o igual al número de núcleos disponibles en el servidor, ya se trate de un equipo virtual o físico. Por ejemplo, si su equipo servidor tiene ocho núcleos, debe comprar una licencia para ocho núcleos como mínimo.

Si usa un equipo servidor con gran cantidad de núcleos CPU, 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 será menos rápida que si se utilizaran todos los núcleos disponibles en el servidor.

2.4 Catálogos XML

El mecanismo de catalogación XML permite recuperar archivos de carpetas locales, lo cual incrementa la velocidad global de procesamiento y mejora la portabilidad de los documentos (porque solo se tienen que cambiar los identificadores URI de los archivos de catálogo). Para más información consulte el apartado Cómo funcionan los catálogos.

Las herramientas XML de Altova usan un mecanismo de catalogación para acceder rápidamente a los archivos más utilizados, como esquemas XML y DTD. El usuario puede personalizar y ampliar este mecanismo de catalogación, que se describe en el apartado Mecanismo de catalogación XML de Altova. En el apartado Variables para ubicaciones del sistema se enumeran variables Windows para las ubicaciones más corrientes. Estas variables se pueden usar en los archivos de catálogo para encontrar las carpetas más utilizadas.

Esta sección se divide en varios apartados:

- Cómo funcionan los catálogos
- Mecanismo de catalogación XML de Altova
- Variables para ubicaciones de sistemas Windows

Para más información consulte la especificación XML Catalogs.

© 2013 Altova GmbH

2.4.1 Cómo funcionan los catálogos

Temas de este apartado:

- Asignar identificadores públicos y de sistema a direcciones URL locales
- Asignar rutas de acceso, direcciones URL web y nombres a direcciones URL locales

---

La función de los catálogos es redireccionar llamadas a recursos remotos a una URL local. Esto se consigue mediante asignaciones en el archivo de catálogo entre identificadores públicos o de sistemas, identificadores URI o partes de identificadores y la URL local correspondiente.

Asignaciones entre identificadores públicos y de sistema y URL locales
Durante la lectura de la declaración DOCTYPE de una DTD en un archivo XML, el identificador público o de sistema de la declaración encuentra el recurso necesario. Si el identificador selecciona un recurso remoto o si el identificador no es un localizador, entonces se puede asignar a un recurso local mediante una entrada en el catálogo.

Por ejemplo, este archivo SVG:

```
<?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>
  ...
</svg>
```

Su identificador público es: **`-//W3C//DTD SVG 1.1//EN`**
Su identificador de sistema es: `http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd`

En el catálogo se puede añadir una entrada para asignar el identificador público a una URL local. Por ejemplo:

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

O una entrada para asignar el identificador de sistema a una URL local. Por ejemplo:

```
<system systemId="http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd" uri="
schemas/svg/svg11.dtd"/>
```

Si se usa el identificador público o de sistema que aparece en el catálogo, entonces se usa la URL a la que está asignado. Las rutas relativas se resuelven con referencia a un atributo `xml:base` en el elemento de redirección del catálogo. La URL base de reserva es la URL del archivo de catálogo. Por el contrario, si no se usa el identificador público o de sistema que aparece en el catálogo, entonces se usará la URL del documento XML (en nuestro ejemplo sería la URL `http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd`).

Asignaciones entre rutas de archivo relativas/absolutas, direcciones URL o nombres y URL locales
El elemento **`uri`** se puede usar para asignar una ruta de archivo relativa/absoluta, una

dirección URL o un nombre a una URL local. Por ejemplo:

```
<uri name="doc.xslt" uri="C:\Docs\doc.xslt"/>
<uri name="U:\Docs\2013\doc.xslt" uri="C:\Docs\doc.xslt"/>
<uri name="http://www.altova.com/schemas/doc.xslt" uri="
C:\Docs\doc.xslt"/>
<uri name="foo" uri="C:\Docs\doc.xslt"/>
```

Cuando se encuentra el valor de `name`, este se asigna al recurso especificado en el atributo `uri`. Con un catálogo distinto, el mismo nombre se podría asignar a un recurso diferente. Por ejemplo:

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

Por lo general, la parte URI del valor del atributo (en negrita) 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 la parte URI del valor del atributo `xsi:schemaLocation` (en vez de `Orgchart.xsd`). El esquema está ubicado dentro del catálogo gracias a la parte de espacio de nombres del valor del atributo `xsi:schemaLocation`. En el ejemplo anterior la parte de espacio de nombres es `http://www.altova.com/schemas/orgchart`.

En el catálogo la entrada siguiente encontraría el esquema por la parte de espacio de nombres.

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

Para más información sobre estos elementos consulte la especificación XML Catalogs.

 © 2013 Altova GmbH

2.4.2 Mecanismo de catalogación XML de Altova

Temas de este apartado:

- El archivo de catálogo raíz `RootCatalog.xml` contiene los archivos de catálogo en los que busca RaptorXML.
- Los archivos catálogo de extensión `CoreCatalog.xml`, `CustomCatalog.xml` y `Catalog.xml`.
- Subconjunto de catálogos compatible.

RootCatalog.xml
RaptorXML busca por defecto en el archivo `RootCatalog.xml` (*ver más abajo*) la lista de archivos de catálogo que debe usar. El catálogo raíz `RootCatalog.xml` está en esta carpeta:

```
<CarpetaArchivosPrograma>\Altova\RaptorXMLServer2013\etc
```

Para usar otro archivo como catálogo raíz, utilice la opción `--catalog` de la línea de comandos, el método `setCatalog` de la interfaz Java o el método `Catalog` de la interfaz COM.

```
<?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"/>
  <nextCatalog catalog="CoreCatalog.xml"/>

  <!-- Incluir todos los catálogos situados en la carpeta Schemas del primer
nivel de directorios -->
  <nextCatalog spy:recurseFrom="%AltovaCommonFolder%/Schemas" catalog="
catalog.xml" spy:depth="1"/>

  <!-- Incluir todos los catálogos situados en la carpeta XBRL del primer
nivel de directorios -->
  <nextCatalog spy:recurseFrom="%AltovaCommonFolder%/XBRL" catalog="
catalog.xml" spy:depth="1"/>
</catalog>
```

El resto de archivos de catálogo se enumeran dentro de un elemento `nextCatalog` y puede añadir tantos archivos como quiera. RaptorXML busca en todos los archivos de catálogo y resuelve las asignaciones que hay en ellos.

En el fragmento de código anterior puede observar una referencia directa a dos catálogos: `CoreCatalog.xml` y `CustomCatalog.xml`. Además se hace referencia a los catálogos llamados `catalog.xml` que están en el primer nivel de subcarpetas de las carpetas `Schemas` y `XBRL`. (El valor de la variable `%AltovaCommonFolder%` se explica en el apartado Variables para ubicaciones de sistema.)

Los archivos de catálogo de `Altova Common Folder` asignan los identificadores públicos y de sistema predefinidos de los esquemas más utilizados (como XML Schema y XHTML) a identificadores URI que apuntan a las copias locales de los esquemas correspondientes. Estos

esquemas se instalan en la carpeta `Altova Common Folder` durante la instalación de RaptorXML.

CoreCatalog.xml, CustomCatalog.xml y Catalog.xml
Los archivos de catálogo `CoreCatalog.xml` y `CustomCatalog.xml` se enumeran en `RootCatalog.xml`:

- `CoreCatalog.xml` contiene ciertas asignaciones propias de Altova necesarias para encontrar esquemas en la carpeta `Altova Common Folder`.
- `CustomCatalog.xml` es un archivo esqueleto donde puede crear sus propias asignaciones. En `CustomCatalog.xml` puede crear asignaciones para cualquier esquema que necesite y que no esté en los archivos de catálogo de la carpeta `Altova Common Folder`. Para ello debe utilizar elementos compatibles del mecanismo de catalogación OASIS (*ver más abajo*).
- Hay varios archivos `Catalog.xml` dentro de las carpetas de esquemas o taxonomías XBRL de la carpeta `Altova Common Folder` y cada uno de estos archivos asigna identificadores públicos/de sistema a identificadores URI que apuntan a copias locales de los esquemas correspondientes.

Tanto `CoreCatalog.xml` como `CustomCatalog.xml` están en la carpeta `<CarpetaArchivosPrograma>\Altova\RaptorXMLServer2013\etc`. Los archivos `catalog.xml` están en las carpetas de esquema correspondientes, situadas a su vez dentro de las carpetas `%AltovaCommonFolder%\Schemas` y `%AltovaCommonFolder%\XBRL`.

Subconjunto de catálogos compatible
Cuando cree entradas en un archivo de catálogo utilizado por RaptorXML, solamente debería usar los elementos de la especificación OASIS que aparecen a continuación. Consulte la especificación XML Catalogs para obtener más información.

- `<public publicId="IDPúblico del Recurso" uri="URL del archivo local"/>`
- `<system systemId="IDdeSistema del Recurso" uri="URL del archivo local "/>`
- `<uri name="nombreArchivo" uri="URL del archivo identificado por el nombre de archivo"/>`
- `<rewriteURI uriStartString="InicioDeCadena del URI que se debe rescribir" rewritePrefix="Cadena que debe sustituir a InicioDeCadena"/>`
- `<rewriteSystem systemIdStartString="InicioDeCadena del IDdeSistema" rewritePrefix="Cadena de sustitución para encontrar el recurso localmente"/>`

Cuando no exista un identificador público, puede asignar el identificador de sistema directamente a una URL con ayuda del elemento `system`. También puede asignar un URI a otro URI usando el elemento `uri`. Los elementos `rewriteURI` y `rewriteSystem` sirven para rescribir la parte inicial de un URI o de un identificador de sistema, respectivamente. Esto permite reemplazar el inicio de una ruta de archivo y, por tanto, apuntar a otro directorio.

Nota: todos los elementos pueden tomar el atributo `xml:base`, que se usa para especificar el URI base del elemento. Si no hay ningún elemento con `xml:base`, el URI base será el URI del archivo de catálogo.

 © 2013 Altova GmbH

Para obtener más información sobre estos elementos consulte la especificación XML Catalogs.

2.4.3 Variables para ubicaciones de sistemas Windows

En los archivos de catálogo puede usar variables de entorno Shell para señalar la ruta de acceso a varias ubicaciones del sistema Windows. Estas son las variables de entorno Shell compatibles:

%AltovaCommonFolder%	C:\Archivos de programa\Altova\Common2013
%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 del directorio de archivos que contiene datos del programa de todos los usuarios.
%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.

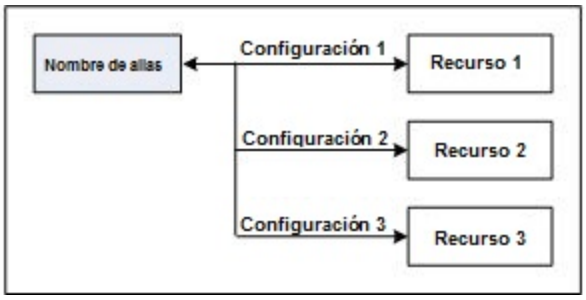
© 2013 Altova GmbH

2.5 Recursos globales

Temas de este apartado:

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

¿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`**.

 © 2013 Altova GmbH

2.6 Problemas de seguridad

Temas de este apartado:

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

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

La interfaz HTTP 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 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` 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. 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` del archivo de configuración del servidor 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 (que es un subdirectorio de `output-root-directory`), 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 para evitar problemas de seguridad.

Altova RaptorXML 2013

Interfaz de la línea de comandos (ILC)

3 Interfaz de la línea de comandos (ILC)

El ejecutable de RaptorXML que sirve para trabajar con la interfaz de la línea de comandos (ILC) se instala por defecto en esta carpeta:

```
<CarpetaArchivosPrograma>\Altova\RaptorXMLServer2013\bin\raptorxml.exe
```

Uso
Esta es la sintaxis de la línea de comandos:

```
RaptorXML --h | --help | --version | <comando> [opciones] [argumentos]
```

`RaptorXML`	Llama a la aplicación
`--h | --help`	Muestra el texto de ayuda
`--version`	Muestra el número de versión de la aplicación
`<comando>`	Comando que se debe ejecutar (*ver la lista que aparece más abajo*). Cada comando (con sus argumentos y opciones respectivos) se describe y explica en los apartados de esta sección.
`[opciones]`	Opciones de un comando. Cada apartado de esta sección describe un comando y sus opciones. Además, el apartado Opciones enumera todas las opciones de la línea de comandos.
`[argumentos]`	Argumentos de un comando. Cada apartado de esta sección describe un comando y sus argumentos.

Comandos de la ILC
A continuación enumeramos todos los comandos de la ILC de RaptorXML ordenados según su función (algunos aparecen en más de una categoría). Estos comandos se describen uno a uno en los apartados de esta sección, junto con sus opciones y argumentos.

Comandos de validación

valdtd | dtd	Valida un documento DTD.
valxml-withdtd | xml	Valida un documento XML con una DTD.
valxml-withxsd | xsi	Valida un documento XML con un esquema XML.
valxquery	Valida un documento XQuery.
valxsd | xsd	Valida un esquema XML del W3C.
valxslt	Valida un documento XSLT.
valany	Valida cualquier tipo de documento de los mencionados anteriormente. El tipo de documento se detecta de forma automática.

© 2013 Altova GmbH

Comandos para comprobar el formato XML

wfxml	Comprueba si un documento XML tiene un formato correcto.
wfdtd	Comprueba si un documento DTD tiene un formato correcto.
wfany	Comprueba si el documento (ya sea XML o DTD) tiene un formato correcto.

Comandos XSLT

xslt	Realiza una transformación con un archivo XSLT dado.
valxslt	Valida un documento XSLT.

Comandos XQuery

xquery	Ejecuta un XQuery con un archivo XQuery dado.
valxquery	Valida un documento XQuery.

3.1 Comandos para validar XML, DTD, XSD

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

- *XML:* los documentos de instancia XML se validan con una DTD (valxml-withdtd | xml) o con un esquema XML 1.0/1.1 (valxml-withxsd | xsi).
- *DTD:* los DTD se revisan para ver si su formato correcto y confirmar que no tienen errores (valdtd | dtd).
- *XSD:* los esquemas XML del W3C (XSD) se validan según las reglas de la especificación XML Schema (valxsd | xsd).

valxml-withdtd | xml	Valida un documento de instancia XML con una DTD.
valxml-withxsd | xsi	Valida un documento de instancia XML con un esquema XML.
valdtd | dtd	Valida un documento DTD.
valxsd | xsd	Valida un documento de esquema XML del W3C (XSD).
valany	Valida cualquier documento, ya sea XML, DTD o XSD. Este comando también se usa para validar documentos XBRL (de instancia o taxonomía), XSLT y XQuery. El tipo de documento suministrado se detecta automáticamente.

Nota: también se pueden validar instancias XBRL, taxonomías XBRL, documentos XSLT y documentos XQuery. Estos comandos de validación se describen en estas secciones: Comandos de validación XBRL, Comandos XSLT y Comandos XQuery.

© 2013 Altova GmbH

3.1.1 valxml-withdtd (xml)

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

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`.

Para validar varios documentos a la vez: (i) enumere en la ILC los archivos que desea validar, separados con un espacio o (ii) enumere en un archivo de texto (`*.txt`) los archivos que desea validar (un nombre de archivo en cada línea) y suministre este archivo de texto con el argumento *ArchivoEntrada* junto con la opción `--listfile` con valor `true` (*ver la lista de opciones más abajo*).

Ejemplos

- **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`

Opciones
Haga clic en una opción o en el título de su categoría para leer su descripción.

- El valor booleano de la opción se establece en `true` si no se especifica ningún valor para la opción.
- El valor de la opción se puede especificar sin comillas en todos los casos excepto en dos: (i) cuando la cadena de valor contiene espacios o (ii) cuando en la descripción de la opción se especifica que las comillas son necesarias.

Opciones del comando valxml-withdtd

`--dtd=`*ARCHIVO*

`--namespaces=true|false`

Opciones de procesamiento

`--listfile=true|false`

`--streaming=true|false`

Opciones de catalogación

`--catalog=`*ARCHIVO*

`--user-catalog=`*ARCHIVO*

Opciones para recursos globales

`--enable-globalresources=true|false`

`--gr, --globalresourcefile=`*ARCHIVO*

`--gc, --globalresourceconfig=`*VALOR*

Opciones para archivos ZIP

--recurse=true|false

Opciones para errores

--error-limit=*N*|ilimitado

--error-format=text|shortxml|longxml

Opciones para mensajes

--log-output=*ARCHIVO*

--verbose=true|false

Opciones de ayuda y versión

--h, --help

--version

© 2013 Altova GmbH

3.1.2 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.

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

El argumento **ArchivoEntrada** suministra el documento XML que debe validarse. La opción --schemalocation-hints=true|false indica si la referencia XSD del documento XML debe utilizarse o no, siendo `true` su valor predeterminado (es decir, se utiliza). La opción --xsd=*ARCHIVO* especifica qué esquema se utiliza.

Para validar varios documentos a la vez: (i) enumere en la ILC los archivos que desea validar, separados con un espacio o (ii) enumere en un archivo de texto (`*.txt`) los archivos que desea validar (un nombre de archivo en cada línea) y suministre este archivo de texto con el argumento *ArchivoEntrada* junto con la opción --listfile con valor `true` (*ver la lista de opciones más abajo*).

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

Ejemplos

- **raptorxml** `valxml-withxsd --schemalocation-hints=false --xsd=c:\MiXSD.xsd c:\SinRefXSD.xml`
- **raptorxml** `xsi c:\SinRefXSD.xml`
- **raptorxml** `xsi --xsd-version=1.1 --listfile=true c:\ListaArchivos.txt`

Opciones
Haga clic en una opción o en el título de su categoría para leer su descripción.

- El valor booleano de la opción se establece en `true` si no se especifica ningún valor para la opción.
- El valor de la opción se puede especificar sin comillas en todos los casos excepto en dos: (i) cuando la cadena de valor contiene espacios o (ii) cuando en la descripción de la opción se especifica que las comillas son necesarias.

Opciones del comando valxml-withxsd

--assessment-mode=skip|lax|strict

Opciones de XML Schema

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

--schema-imports=load-by-schemalocation|
load-preferring-schemalocation|
load-by-namespace|
load-combining-both|
license-namespace-only

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

--xsd=*ARCHIVO*
--xsd-version=1.0|1.1|detect

Opciones para instancias XML

--xinclude=true|false
--xml-mode=wf|id|valid

Opciones de procesamiento

--listfile=true|false
--streaming=true|false
--script=*ARCHIVO*

Opciones de catalogación

--catalog=*ARCHIVO*
--user-catalog=*ARCHIVO*

Opciones para recursos globales

--enable-globalresources=true|false
--gr, --globalresourcefile=*ARCHIVO*
--gc, --globalresourceconfig=*VALOR*

Opciones para archivos ZIP

--recurse=true|false

Opciones para errores

--error-limit=*N*|ilimitado
--error-format=text|shortxml|longxml

Opciones para mensajes

--log-output=*ARCHIVO*
--verbose=true|false

Opciones de ayuda y versión

--h, --help
--version

 © 2013 Altova GmbH

3.1.3 valdtd (dtd)

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

```
raptorxml valdtd | dtd [opciones] ArchivoEntrada
```

El argumento *ArchivoEntrada* es el documento DTD que debe validarse.

Para validar varios documentos a la vez: (i) enumere en la ILC los archivos que desea validar, separados con un espacio o (ii) enumere en un archivo de texto (`*.txt`) los archivos que desea validar (un nombre de archivo en cada línea) y suministre este archivo de texto con el argumento *ArchivoEntrada* junto con la opción `--listfile` con valor `true` (*ver la 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

Opciones
Haga clic en una opción o en el título de su categoría para leer su descripción.

- El valor booleano de la opción se establece en `true` si no se especifica ningún valor para la opción.
- El valor de la opción se puede especificar sin comillas en todos los casos excepto en dos: (i) cuando la cadena de valor contiene espacios o (ii) cuando en la descripción de la opción se especifica que las comillas son necesarias.

Opciones de procesamiento

`--listfile=true|false`

Opciones de catalogación

`--catalog=ARCHIVO`

`--user-catalog=ARCHIVO`

Opciones para recursos globales

`--enable-globalresources=true|false`

`--gr, --globalresourcefile=ARCHIVO`

`--gc, --globalresourceconfig=VALOR`

Opciones para archivos ZIP

`--recurse=true|false`

Opciones para errores

`--error-limit=N|ilimitado`

--error-format=text|shortxml|longxml

Opciones para mensajes

--log-output=*ARCHIVO*

--verbose=true|false

Opciones de ayuda y versión

--h, --help

--version

 © 2013 Altova GmbH

3.1.4 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.

```
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 indica la versión XSD con la que debe validarse, siendo 1.0 su valor predeterminado.

Para validar varios documentos a la vez: (i) enumere en la ILC los archivos que desea validar, separados con un espacio o (ii) enumere en un archivo de texto (*.txt) los archivos que desea validar (un nombre de archivo en cada línea) y suministre este archivo de texto con el argumento *ArchivoEntrada* junto con la opción --listfile con valor true (*ver la lista de opciones más abajo*).

Ejemplos

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

Opciones
Haga clic en una opción o en el título de su categoría para leer su descripción.

- El valor booleano de la opción se establece en true si no se especifica ningún valor para la opción.
- El valor de la opción se puede especificar sin comillas en todos los casos excepto en dos: (i) cuando la cadena de valor contiene espacios o (ii) cuando en la descripción de la opción se especifica que las comillas son necesarias.

Opciones de XML Schema

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

--schema-imports=load-by-schemalocation|
load-preferring-schemalocation|
load-by-namespace|
load-combining-both|
license-namespace-only

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

--xsd-version=1.0|1.1|detect

Opciones de procesamiento

--listfile=true|false

--script=*ARCHIVO*

Opciones de catalogación

--catalog=*ARCHIVO*

--user-catalog=*ARCHIVO*

Opciones para recursos globales

--enable-globalresources=true|false

--gr, --globalresourcefile=*ARCHIVO*

--gc, --globalresourceconfig=*VALOR*

Opciones para archivos ZIP

--recurse=true|false

Opciones para instancias XML

--xinclude=true|false

--xml-mode=wf|id|valid

Opciones para errores

--error-limit=*N*|ilimitado

--error-format=text|shortxml|longxml

Opciones para mensajes

--log-output=*ARCHIVO*

--verbose=true|false

Opciones de ayuda y versión

--h, --help

--version

 © 2013 Altova GmbH

3.1.5 valany

El comando **valany** valida un documento XML, DTD o esquema XML con sus especificaciones correspondientes. El tipo de documento se detecta automáticamente.

```
raptorxml valany [opciones] ArchivoEntrada
```

El argumento *ArchivoEntrada* suministra el documento que debe validarse. Recuerde que como argumento del comando se puede suministrar solamente un documento. El tipo de documento se detecta automáticamente.

Ejemplos

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

Opciones
Haga clic en una opción o en el título de su categoría para leer su descripción.

- El valor booleano de la opción se establece en `true` si no se especifica ningún valor para la opción.
- El valor de la opción se puede especificar sin comillas en todos los casos excepto en dos: (i) cuando la cadena de valor contiene espacios o (ii) cuando en la descripción de la opción se especifica que las comillas son necesarias.

Opciones de XML Schema

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

--schema-imports=load-by-schemalocation|
load-preferring-schemalocation|
load-by-namespace|
load-combining-both|
license-namespace-only

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

Opciones de catalogación

--catalog=*ARCHIVO*

--user-catalog=*ARCHIVO*

Opciones para recursos globales

--enable-globalresources=true|false

--gr, --globalresourcefile=*ARCHIVO*

--gc, --globalresourceconfig=*VALOR*

Opciones para archivos ZIP

--recurse=true|false

Opciones para errores

--error-limit=*N*|ilimitado

--error-format=text|shortxml|longxml

Opciones para mensajes

--log-output=*ARCHIVO*

--verbose=true|false

Opciones de ayuda y versión

--h, --help

--version

© 2013 Altova GmbH

3.2 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	Comprueba si el documento XML tiene un formato correcto.
wfdtd	Comprueba si el documento DTD tiene un formato correcto.
wfany	Comprueba si el documento XML o DTD tiene un formato correcto. El tipo se detecta automáticamente.

3.2.1 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.

```
raptorxml wfxml [opciones] ArchivoEntrada
```

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

Para comprobar el formato de varios documentos: (i) enumere en la ILC los archivos que desea revisar, separados con un espacio o (ii) enumere en un archivo de texto (`*.txt`) los archivos que desea revisar (un nombre de archivo en cada línea) y suministre este archivo de texto con el argumento *ArchivoEntrada* junto con la opción `--listfile` con valor `true` (*ver la 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

Opciones
Haga clic en una opción o en el título de su categoría para leer su descripción.

- El valor booleano de la opción se establece en `true` si no se especifica ningún valor para la opción.
- El valor de la opción se puede especificar sin comillas en todos los casos excepto en dos: (i) cuando la cadena de valor contiene espacios o (ii) cuando en la descripción de la opción se especifica que las comillas son necesarias.

Opciones de procesamiento

--listfile=true|false
--streaming=true|false
--dtd=*ARCHIVO*
--namespaces=true|false

Opciones de catalogación

--catalog=*ARCHIVO*
--user-catalog=*ARCHIVO*

Opciones para recursos globales

--enable-globalresources=true|false
--gr, --globalresourcefile=*ARCHIVO*
--gc, --globalresourceconfig=*VALOR*

Opciones para archivos ZIP

--recurse=true|false

 © 2013 Altova GmbH

Opciones para errores

--error-limit=*N*|ilimitado

--error-format=text|shortxml|longxml

Opciones para mensajes

--log-output=*ARCHIVO*

--verbose=true|false

Opciones de ayuda y versión

--h, --help

--version

3.2.2 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.

```
raptorxml wfdtd [opciones] ArchivoEntrada
```

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

Para comprobar el formato de varios documentos: (i) enumere en la ILC los archivos que desea revisar, separados con un espacio o (ii) enumere en un archivo de texto (`*.txt`) los archivos que desea revisar (un nombre de archivo en cada línea) y suministre este archivo de texto con el argumento *ArchivoEntrada* junto con la opción `--listfile` con valor `true` (*ver la 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

Opciones
Haga clic en una opción o en el título de su categoría para leer su descripción.

- El valor booleano de la opción se establece en `true` si no se especifica ningún valor para la opción.
- El valor de la opción se puede especificar sin comillas en todos los casos excepto en dos: (i) cuando la cadena de valor contiene espacios o (ii) cuando en la descripción de la opción se especifica que las comillas son necesarias.

Opciones de procesamiento
`--listfile=true|false`

Opciones de catalogación
`--catalog=ARCHIVO`
`--user-catalog=ARCHIVO`

Opciones para recursos globales
`--enable-globalresources=true|false`
`--gr, --globalresourcefile=ARCHIVO`
`--gc, --globalresourceconfig=VALOR`

Opciones para archivos ZIP
`--recurse=true|false`

Opciones para errores
`--error-limit=N|ilimitado`

 © 2013 Altova GmbH

--error-format=text|shortxml|longxml

Opciones para mensajes

--log-output=*ARCHIVO*

--verbose=true|false

Opciones de ayuda y versión

--h, --help

--version

3.2.3 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.

```
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

Opciones
Haga clic en una opción o en el título de su categoría para leer su descripción.

- El valor booleano de la opción se establece en `true` si no se especifica ningún valor para la opción.
- El valor de la opción se puede especificar sin comillas en todos los casos excepto en dos: (i) cuando la cadena de valor contiene espacios o (ii) cuando en la descripción de la opción se especifica que las comillas son necesarias.

Opciones de catalogación

--catalog=*ARCHIVO*

--user-catalog=*ARCHIVO*

Opciones para recursos globales

--enable-globalresources=true|false

--gr, --globalresourcefile=*ARCHIVO*

--gc, --globalresourceconfig=*VALOR*

Opciones para archivos ZIP

--recurse=true|false

Opciones para errores

--error-limit=*N*|ilimitado

--error-format=text|shortxml|longxml

Opciones para mensajes

--log-output=*ARCHIVO*

--verbose=true|false

Opciones de ayuda y versión

--h, --help

--version

 © 2013 Altova GmbH

3.3 Comandos XSLT

Estos son los comandos XSLT:

- xslt: para transformar documentos XML con un documento XSLT
- valxslt: para validar documentos XSLT

Los argumentos y opciones de cada comando se describen en los siguientes apartados.

© 2013 Altova GmbH

3.3.1 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.

```
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) o el punto de entrada de una plantilla con nombre (--template-entry-point). Si no especifica la opción del documento de salida --output, 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/@att1 --p=title:'cadenasinespacios' --param=title:"'cadena con espacios'" --p=amount:456 c:\Test.xslt

---

Opciones

Haga clic en una opción o en el título de su categoría para leer su descripción.

- El valor booleano de la opción se establece en `true` si no se especifica ningún valor para la opción.
- El valor de la opción se puede especificar sin comillas en todos los casos excepto en dos: (i) cuando la cadena de valor contiene espacios o (ii) cuando en la descripción de la opción se especifica que las comillas son necesarias.

Opciones propias del comando xslt

--indent-characters=*VALOR*

--input=*ARCHIVO*

--output=*ARCHIVO*

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

--streaming-serialization-enabled=true|false

Opciones compartidas por los comandos xslt y valxslt

--chartext-disable=true|false

--dotnetext-disable=true|false

--javaext-barcode-location=*ARCHIVO*

--javaext-disable=true|false

--template-entry-point=*VALOR*

--template-mode=*VALOR*

--xslt-version=1|2|3

Opciones de catalogación

--catalog=*ARCHIVO*

--user-catalog=*ARCHIVO*

Opciones para recursos globales

--enable-globalresources=true|false

--gr, --globalresourcefile=*ARCHIVO*

--gc, --globalresourceconfig=*VALOR*

Opciones para errores

--error-limit=*N*|ilimitado

Opciones para mensajes

--verbose=true|false

Opciones para instancias XML

--xinclude=true|false

--xml-mode=wf|id|valid

Opciones de XML Schema

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

--schema-imports=load-by-schemalocation|
load-preferring-schemalocation|
load-by-namespace|
load-combining-both|
license-namespace-only

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

--xsd-version=1.0|1.1|detect

Opciones de ayuda y versión

--h, --help

--version

© 2013 Altova GmbH

3.3.2 valxslt

El comando **valxslt** toma un archivo XSLT como único argumento y lo valida.

```
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

Opciones
Haga clic en una opción o en el título de su categoría para leer su descripción.

- El valor booleano de la opción se establece en `true` si no se especifica ningún valor para la opción.
- El valor de la opción se puede especificar sin comillas en todos los casos excepto en dos: (i) cuando la cadena de valor contiene espacios o (ii) cuando en la descripción de la opción se especifica que las comillas son necesarias.

Opciones compartidas por los comandos xslt y valxslt

--chartext-disable=true|false
--dotnetext-disable=true|false
--javaext-barcode-location=*ARCHIVO*
--javaext-disable=true|false
--template-entry-point=*VALOR*
--template-mode=*VALOR*
--xslt-version=1|2|3

Opciones de catalogación

--catalog=*ARCHIVO*
--user-catalog=*ARCHIVO*

Opciones para recursos globales

--enable-globalresources=true|false
--gr, --globalresourcefile=*ARCHIVO*
--gc, --globalresourceconfig=*VALOR*

Opciones para errores

--error-limit=*N*|ilimitado

Opciones para mensajes

--verbose=true|false

Opciones para instancias XML

--xinclude=true|false

--xml-mode=wf|id|valid

Opciones de XML Schema

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

--schema-imports=load-by-schemalocation|
load-preferring-schemalocation|
load-by-namespace|
load-combining-both|
license-namespace-only

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

--xsd-version=1.0|1.1|detect

Opciones de ayuda y versión

--h, --help

--version

© 2013 Altova GmbH

3.4 Comandos XQuery

Estos son los comandos XQuery:

- xquery: para ejecutar documentos XQuery y, si se quiere, con un documento de entrada
- valxquery: para validar documentos XQuery

Los argumentos y opciones de cada comando se describen en los siguientes apartados.

3.4.1 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.

```
raptorxml xquery [opciones] Archivo-XQuery
```

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

Ejemplos

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

---

Opciones
Haga clic en una opción o en el título de su categoría para leer su descripción.

- El valor booleano de la opción se establece en `true` si no se especifica ningún valor para la opción.
- El valor de la opción se puede especificar sin comillas en todos los casos excepto en dos: (i) cuando la cadena de valor contiene espacios o (ii) cuando en la descripción de la opción se especifica que las comillas son necesarias.

***Opciones propias del comando* xquery**

--indent-characters=*VALOR*
--input=*ARCHIVO*
--output=*ARCHIVO*
--output-encoding=*VALOR*
--output-indent=true|false
--output-method=xml|html|xhtml|text
--p, --param=*CLAVE:VALOR*

***Opciones compartidas por los comandos* xquery *y* valxquery**

--omit-xml-declaration=true|false
--xquery-version=1|3

Opciones de catalogación

--catalog=*ARCHIVO*
--user-catalog=*ARCHIVO*

Opciones para recursos globales

 © 2013 Altova GmbH

--enable-globalresources=true|false
--gr, --globalresourcefile=*ARCHIVO*
--gc, --globalresourceconfig=*VALOR*

Opciones para errores
--error-limit=*N*|ilimitado

Opciones para mensajes
--verbose=true|false

Opciones para instancias XML
--xinclude=true|false
--xml-mode=wf|id|valid

Opciones de XML Schema
--xsd-version=1.0|1.1|detect

Opciones de ayuda y versión
--h, --help
--version

3.4.2 valxquery

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

```
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

Opciones
Haga clic en una opción o en el título de su categoría para leer su descripción.

- El valor booleano de la opción se establece en `true` si no se especifica ningún valor para la opción.
- El valor de la opción se puede especificar sin comillas en todos los casos excepto en dos: (i) cuando la cadena de valor contiene espacios o (ii) cuando en la descripción de la opción se especifica que las comillas son necesarias.

***Opciones compartidas por los comandos* xquery *y* valxquery**

--omit-xml-declaration=true|false

--xquery-version=1|3

Opciones de catalogación

--catalog=*ARCHIVO*

--user-catalog=*ARCHIVO*

Opciones para recursos globales

--enable-globalresources=true|false

--gr, --globalresourcefile=*ARCHIVO*

--gc, --globalresourceconfig=*VALOR*

Opciones para errores

--error-limit=*N*|ilimitado

Opciones para mensajes

--verbose=true|false

Opciones para instancias XML

--xinclude=true|false

--xml-mode=wf|id|valid

Opciones de XML Schema

--xsd-version=1.0|1.1|detect

 © 2013 Altova GmbH

Opciones de ayuda y versión

--h, --help

--version

3.5 Comandos de ayuda y licencias

En esta sección describimos dos características importantes de RaptorXML:

- Ayuda: cómo obtener información sobre los comandos disponibles o sobre los argumentos y opciones de un comando.
- Licencias: cómo asignar licencias a RaptorXML.

© 2013 Altova GmbH

3.5.1 Ayuda

El comando **help** toma un solo argumento: el nombre del comando para el que desea obtener información adicional. Muestra la sintaxis del comando y otra información importante para poder ejecutar el comando correctamente.

```
raptorxml help Comando
```

Nota: si ejecuta el comando **help** sin ningún argumento, aparece la ayuda sobre todos los comandos de la ILC, cada uno con una breve descripción.

Ejemplo

```
RaptorXML help valany
```

Este comando contiene un argumento, el comando `valany`. Cuando se ejecuta este comando, aparece información de ayuda sobre el comando `valany`.

La opción `--help`
También puede ver la información de ayuda de un comando si usa la opción `--help` con dicho comando. Por ejemplo, si usamos la opción `--help` con el comando `valany`:

```
RaptorXML valany --help
```

conseguimos el mismo resultado que cuando usamos el comando `help` con el argumento `valany`:

```
RaptorXML help valany
```

En ambos casos aparece información de ayuda sobre el comando `valany`.

3.5.2 Licencias

El comando `licenseserver` registra RaptorXMLServer con el servidor de licencias Altova LicenseServer. Este comando toma como argumento el nombre o la dirección IP del servidor en el que se ejecuta LicenseServer.

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

Tras registrar RaptorXML con LicenseServer, se devuelve la URL de la interfaz web de LicenseServer. Escriba la URL en una ventana del explorador para acceder a la interfaz web y después siga los pasos descritos en la documentación de LicenseServer para asignar una licencia a RaptorXML.

Ejemplo

```
RaptorXML licenseserver DOC.altova.com
```

El comando especifica que el equipo `DOC.altova.com` es el equipo donde se ejecuta el servidor de licencias LicenseServer.

Opciones
Las opciones de este comando son:

`--j|json=true|false`
Imprime el resultado del intento de registro en formato de objeto JSON analizable por computador.

`--h|help`
Muestra el texto de ayuda del comando.

`--version`
Muestra el número de versión de RaptorXML. Esta opción debe escribirse antes del comando: `RaptorXML --version licenseserver`.

 © 2013 Altova GmbH

3.6 Comandos de localización

Si quiere puede localizar (traducir) la aplicación RaptorXML en cualquier idioma. RaptorXML ya está disponible en cuatro idiomas (español, inglés, alemán y japonés) en la carpeta `<CarpetaArchivosPrograma>\Altova\RaptorXMLServer2013\bin\`.

Para localizar RaptorXML en otros idiomas:

1. Genere un archivo XML con las cadenas de recursos usando el comando `exportresourcestrings`. Las cadenas de recursos del archivo XML generado estarán en uno de estos idiomas: inglés (`en`), español (`es`), 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\RaptorXMLServer2013\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` para establecer el archivo DLL localizado como aplicación predeterminada. Use el código de idioma del nombre del archivo DLL como argumento del comando `setdeflang`.

Nota: Altova RaptorXML Server está disponible en cuatro idiomas (español, 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`.

3.6.1 exportresourcestrings

El comando **exportresourcestrings** genera un archivo XML que contiene las cadenas de recursos de RaptorXML. Este comando toma dos argumentos: (i) el idioma de las cadenas de recursos del archivo XML de salida y (ii) la ruta de acceso y el nombre del archivo XML de salida. Los idiomas de exportación compatibles (y sus códigos) son: inglés (en), alemán, (de), español (es) y japonés (ja).

```
raptorxml exportresourcestrings CódigoIdioma ArchivoXMLSalida
```

Nota: en los sistemas Linux es necesario usar **raptorxml** todo en minúsculas para llamar al ejecutable.

Argumentos
El comando exportresourcestrings toma dos argumentos:

CódigoIdioma	idioma de las cadenas de recursos del archivo XML de salida exportado. Idiomas permitidos: en, de, es, ja
ArchivoXMLSalida	ubicación y nombre del archivo XML de salida exportado.

Ejemplo
Este comando crea un archivo llamado Strings.xml en c:\ que contiene todas las cadenas de recursos de la aplicación RaptorXML traducida al alemán.

```
raptorxml exportresourcestrings de c:\Strings.xml
```

 © 2013 Altova GmbH

3.6.2 setdeflang

El comando **setdeflang** (formato abreviado **sdl**) establece el idioma predeterminado de RaptorXML. Su argumento obligatorio es `CódigoIdioma`.

```
FlowForceServer setdeflang | sdl CódigoIdioma
```

Nota: en los sistemas Linux es necesario usar **raptorxml** todo en minúsculas para llamar al ejecutable.

Ejemplo
Este comando define el idioma español como idioma predeterminado de los mensajes de la aplicación.

```
raptorxml setdeflang es
```

Idiomas compatibles
Los idiomas disponibles son:

EN	inglés
DE	alemán
ES	español
JA	japonés

3.7 Opciones

En esta sección describimos todas las opciones de la ILC, organizadass por funciones. Para ver qué opción debe usar con cada comando, consulte la descripción del comando correspondiente.

- Catálogos
- Errores
- Recursos globales
- Ayuda y versión
- Mensajes
- Procesamiento
- Evaluación XBRL
- Esquemas XBRL
- Documento XML
- Validación XML
- Esquemas XML (XSD)
- XQuery
- XSLT
- Archivos ZIP

© 2013 Altova GmbH

3.7.1 Catálogos

[--catalog, --user-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.

--user-catalog=*ARCHIVO*
Especifica la ruta de acceso absoluta a un catálogo XML que debe utilizarse junto con el catálogo raíz.

3.7.2 Errores

[--error-limit, --error-format]

--error-limit=*N*|ilimitado
Especifica el límite de errores. Esta opción es útil para limitar el uso del procesador durante la validación. Cuando se alcanza el límite de error, se detiene la validación.
Valor predeterminado: `100`.

--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`.

© 2013 Altova GmbH

3.7.3 Recursos globales

[--enable-global-resources, --globalresourcefile, --globalresourceconfig]

--enable-globalresources=true|false
Habilita la función de recursos globales.
Valor predeterminado: `false`.

--gr, --globalresourcefile=*ARCHIVO*
Especifica el archivo de recursos globales (y habilita los recursos globales).

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

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

3.7.4 Ayuda y versión

[--help, --version]

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

© 2013 Altova GmbH

3.7.5 Mensajes

[--log-output, --verbose]

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

--verbose=true|false
Si el valor es `true`, RaptorXML 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`.

3.7.6 Procesamiento

[--listfile, --script, --streaming-serialization-enabled, --streaming]

--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`

--script=*ARCHIVO*
Una vez finalizada la validación, ejecuta el script Python.

--streaming=true|false
Habilita la validación de 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.
Valor predeterminado: `true`.

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

 © 2013 Altova GmbH

3.7.7 Instancias XML

[--xinclude, --xml-mode]

--xinclude=true|false
Habilita la compatibilidad con inclusiones XML (XInclude). Si el valor es `false`, los elementos XInclude `include` se ignoran.
Valor predeterminado: `false`.

--xml-mode=wf|id|valid
Especifica el modo de procesamiento XML que debe utilizarse: `wf`=comprobación de formato ; `id`=comprobación de formato con ID/IDREF; `valid`=validación.
Valor predeterminado: `wf`.

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

3.7.8 Validación de instancias XML

[--dtd, --xsd, --namespaces, --assessment-mode]

--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.

--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.

--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`.

--assessment-mode=skip|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`.

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

 © 2013 Altova GmbH

3.7.9 Esquemas XML (XSD)

[--schemalocation-hints, --schema-imports, --schema-mapping, --xsd-version]
[Nota sobre schemaLocation-hints]

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

- **Valor predeterminado**: `load-by-schemalocation`. Este valor toma la URL de la ubicación del esquema de los atributos `xsi:schemaLocation` y `xsi:noNamespaceSchemaLocation` de los documentos XML.
- El valor `load-by-namespace` toma la parte de espacio de nombres 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.
- Si usa el valor `load-combining-both` y el espacio de nombres o la URL tienen una asignación de catálogo, se usa dicha asignación. Si ambos tienen asignaciones de catálogo, el valor de la opción `--schema-mapping` 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-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">`. El comportamiento de los atributos se modifica usando estos valores en la opción:

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

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

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, 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). El valor `prefer-schemalocation` es el **valor predeterminado**.

--xsd-version=1.0|1.1|detect

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

El **valor predeterminado** es `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 `version` del elemento `<xs:schema>` del documento. Si el valor del atributo `@version` es `1.1`, se entiende que la versión del esquema es `1.1`. Si el atributo tiene otro valor que no sea `1.1`, se entiende que la versión del esquema es `1.0`. Tenga en cuenta que este mecanismo de detección no está definido en la especificación XML Schema y se utiliza para ser coherentes con el mecanismo de detección de esquemas utilizado en otras aplicaciones de Altova.

Nota sobre la opción `--schemaLocation-hints`
Los documentos de instancia pueden usar sugerencias para indicar la ubicación del esquema. Para ello se usan dos atributos:

- `xsi:schemaLocation` para documentos de esquema con espacios de nombres de destino. El valor del atributo es un par de elementos: el primero es un espacio de nombres y el segundo es una URL que encuentra el esquema. El nombre del espacio de nombres debe coincidir con el espacio de nombres de destino del 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 del atributo es la URL del esquema. El documento de esquema referenciado 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 dos atributos, sobre todo

 © 2013 Altova GmbH

cómo trabajar con la información del atributo `schemaLocation` (*ver la descripción de la opción que aparece más arriba*). Recuerde que RaptorXML Server considera que la parte de espacio de nombres del valor de `xsi:noNamespaceSchemaLocation` es una cadena vacía.

Las sugerencias sobre la ubicación del esquema también se pueden incluir en una instrucción de importación `import` de un esquema XML.

```
<import namespace="unEspacioNombres" schemaLocation="unaURL">
```

En la instrucción de importación `import`, las sugerencias se pueden hacer por medio de un espacio de nombres asignado a un esquema en el archivo de catálogo o con una URL directamente, en el atributo `schemaLocation`. La opción `--schema-imports` especifica cómo se selecciona la ubicación del esquema.

3.7.10 XQuery

Opciones propias del comando `xquery`
[--indent-characters, --input, --output, --output-encoding, --output-indent, --output-method, --param]

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

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

`--output=`*`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. Si no se especifica la opción `--output`, se genera un resultado estándar.

`--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=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`.

`--output-method=xml|html|xhtml|text`
Especifica el formato de salida.
Valor predeterminado: `xml`.

`--p, --param=`*`CLAVE:VALOR`*
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.

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

Opciones compartidas por los comandos `xquery` y `valxquery`
[--omit-xml-declaration, --xquery-version]

 © 2013 Altova GmbH

--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`.

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

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

3.7.11 XSLT

Opciones propias del comando `xslt`
[--indent-characters, --input, --output, --param, --streaming-serialization-enabled]

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

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

`--output=`*`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. Si no se especifica la opción `--output`, se genera un resultado estándar.

`--p, --param=`*`CLAVE:VALOR`*
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:\Salida.xml --param
date=//node/@att1 --p=title:'cadenasinespacios' --param=title:"'cadena con
espacios'" --p=amount:456 c:\Test.xslt
```

`--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`.

Opciones compartidas entre los comandos `xslt` y `valxslt`
[--chartext-disable, --dotnetext-disable, --javaext-barcode-location, --javaext-disable,, --template-entry-point, --template-mode, --xslt-version]

`--chartext-disable=true|false`
Deshabilita las extensiones de gráficos.
Valor predeterminado: `false`
Opción no disponible en RaptorXML Development Edition.

`--dotnetext-disable=true|false`
Deshabilita las extensiones .NET.
Valor predeterminado: `false`

`--javaext-barcode-location=`*`ARCHIVO`*
Especifica la ubicación del archivo de extensión de código de barras

`--javaext-disable=true|false`
Deshabilita las extensiones Java.

 © 2013 Altova GmbH

Valor predeterminado: `false`

`--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.

`--template-mode=`*`VALOR`*
Especifica el modo de plantilla que debe usarse para la transformación.

`--xslt-version=1|2|3`
Especifica si el procesador XSLT debe usar XSLT 1.0, XSLT 2.0 o XSLT 3.0.
Valor predeterminado: `3`

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

3.7.12 Archivos ZIP

[--recurse]

--recurse=true|false

Esta opción se utiliza para seleccionar ficheros dentro de un archivo 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`.

El valor predeterminado de esta opción es `false`.

Opción no disponible en RaptorXML Development Edition.

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

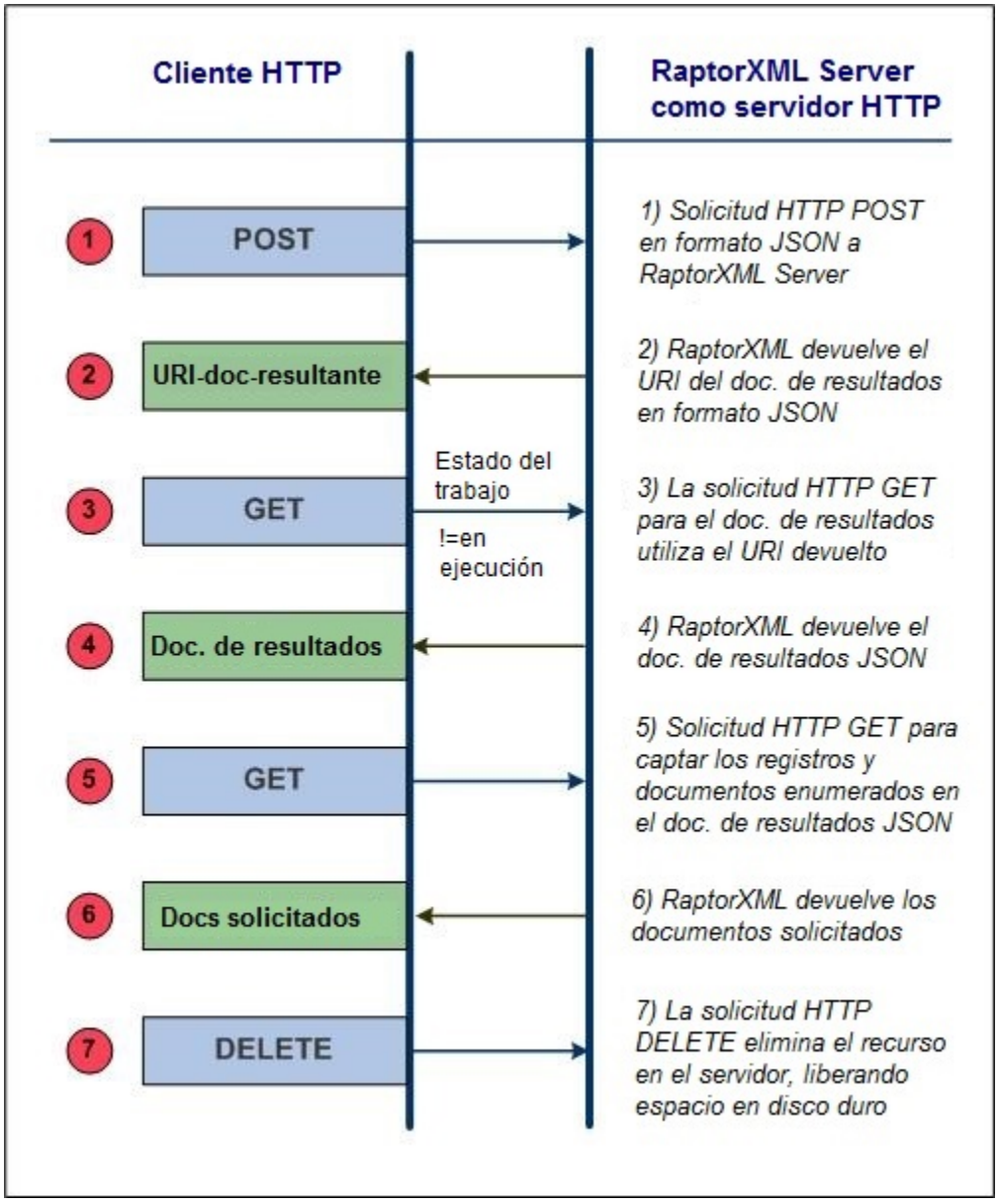
 © 2013 Altova GmbH

Altova RaptorXML 2013

Interfaz HTTP

4 Interfaz HTTP

RaptorXML Server acepta trabajos de validación suministrados por HTTP. La descripción del trabajo así como los resultados se intercambian en formato JSON. El diagrama que aparece a continuación muestra el flujo de trabajo.



 © 2013 Altova GmbH

Problemas de seguridad relacionados con la interfaz HTTP
La interfaz HTTP 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 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` 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, mientras que la sección Solicitudes cliente describe cómo se envían solicitudes clientes al servidor.

4.1 Preparar el servidor

Siga las instrucciones que aparecen a continuación para preparar correctamente RaptorXML Server. Recuerde que antes debe instalarlo y asignarle licencias.

1. Para poder acceder a RaptorXML Server por HTTP antes debe iniciarlo como servicio o como aplicación. Hay varias formas de hacerlo, dependiendo del sistema operativo: en Windows, en Linux, en Mac OS X.
2. Utilice la configuración inicial del servidor para probar la conexión con el servidor. (La configuración inicial del servidor 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, `server_config.xml`. Si quiere cambiar alguna opción de configuración, edite el archivo y guarde los cambios.
4. Si editó el archivo de configuración del servidor, 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.

© 2013 Altova GmbH

4.1.1 Iniciar el servidor

Temas de este apartado:

- Ubicación del ejecutable del servidor
- Iniciar RaptorXML como servicio (Windows)
- Iniciar Raptor XML como servicio (Linux)
- Iniciar RaptorXML como servicio (Mac OS X)

Ubicación del ejecutable del servidor
El ejecutable de RaptorXML Server se instala por defecto en esta carpeta:

```
<CarpetaArchivosProgramas>\Altova\RaptorXMLServer2013\bin\RaptorXMLServer.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` del archivo de configuración del servidor. 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]                  sudo /etc/init.d/raptorxmlserver start
[Ubuntu, CentOS, RedHat]  sudo initctl start raptorxmlserver
```

Si necesita detener RaptorXML Server, utilice este otro comando:

```
[Debian]                  sudo /etc/init.d/raptorxmlserver stop
```

```
[Ubuntu, CentOS, RedHat]   sudo initctl stop raptorxmlserver
```

Iniciar RaptorXML como servicio en Mac OS X

Utilice este comando para iniciar RaptorXML Server como servicio:

```
sudo launchctl load /Library/LaunchDaemons/com.altova.RaptorXMLServer2013
.plist
```

Si necesita detener RaptorXML Server, utilice este otro comando:

```
sudo launchctl unload /Library/LaunchDaemons/com.altova.RaptorXMLServer2013
.plist
```


© 2013 Altova GmbH

4.1.2 Probar la conexión

Temas de este apartado:

- Solicitud GET para probar la conexión
- Respuesta del servidor y ejemplo de datos JSON

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.

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), recomendamos que pruebe la conexión una vez más.

4.1.3 Configurar el servidor

Temas de este apartado:

- Archivo de configuración del servidor: configuración inicial
- Archivo de configuración del servidor: modificar configuración inicial, volver a la configuración inicial
- Archivo de configuración del servidor: ejemplo y opciones de configuración
- Archivo de configuración del servidor: descripción de las opciones de configuración
- Configurar la dirección del servidor

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\RaptorXMLServer2013\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\RaptorXMLServer2013\Output\`.

Las demás opciones de configuración predeterminadas aparecen más abajo en el fragmento 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), 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`:

```
RaptorXMLServer.exe createconfig
```

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.

© 2013 Altova GmbH

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\RaptorXMLServer2013\Output\</
  server.output-root-dir>
  <server.script-root-dir>C:\Program Files (x86)\Altova\RaptorXMLServer2013
  \etc\scripts\</server.script-root-dir>
  <!--<server.catalog-file>catalog.xml</server.catalog-file>-->
  <server.log-file>C:\ProgramData\Altova\RaptorXMLServer2013\Log\server.log</
  server.log-file>
  <http.environment>production</http.environment>
  <!--<http.socket-host>localhost</http.socket-host>-->
  <http.socket-port>8087</http.socket-port>
  <http.log-screen>true</http.log-screen>
  <http.access-file>C:\ProgramData\Altova\RaptorXMLServer2013\Log\access.log</
  http.access-file>
  <http.error-file>C:\ProgramData\Altova\RaptorXMLServer2013\Log\error.log</
  http.error-file>

</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|ja` (inglés, alemán, español y japonés respectivamente). Para aprender a localizar (=traducir) RaptorXML, consulte el apartado Comandos de localización.

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). Si el valor es `false`, los archivos se escriben en el directorio del trabajo del directorio de salida y el URI del archivo se incluye en el documento de resultados. 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 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*.

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\RaptorXMLServer2013\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.

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.

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*) 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.

 © 2013 Altova GmbH

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 servidor tiene la dirección IP `100.60.300.6` 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://100.60.300.6:8087/
```

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).

4.2 Solicitudes cliente

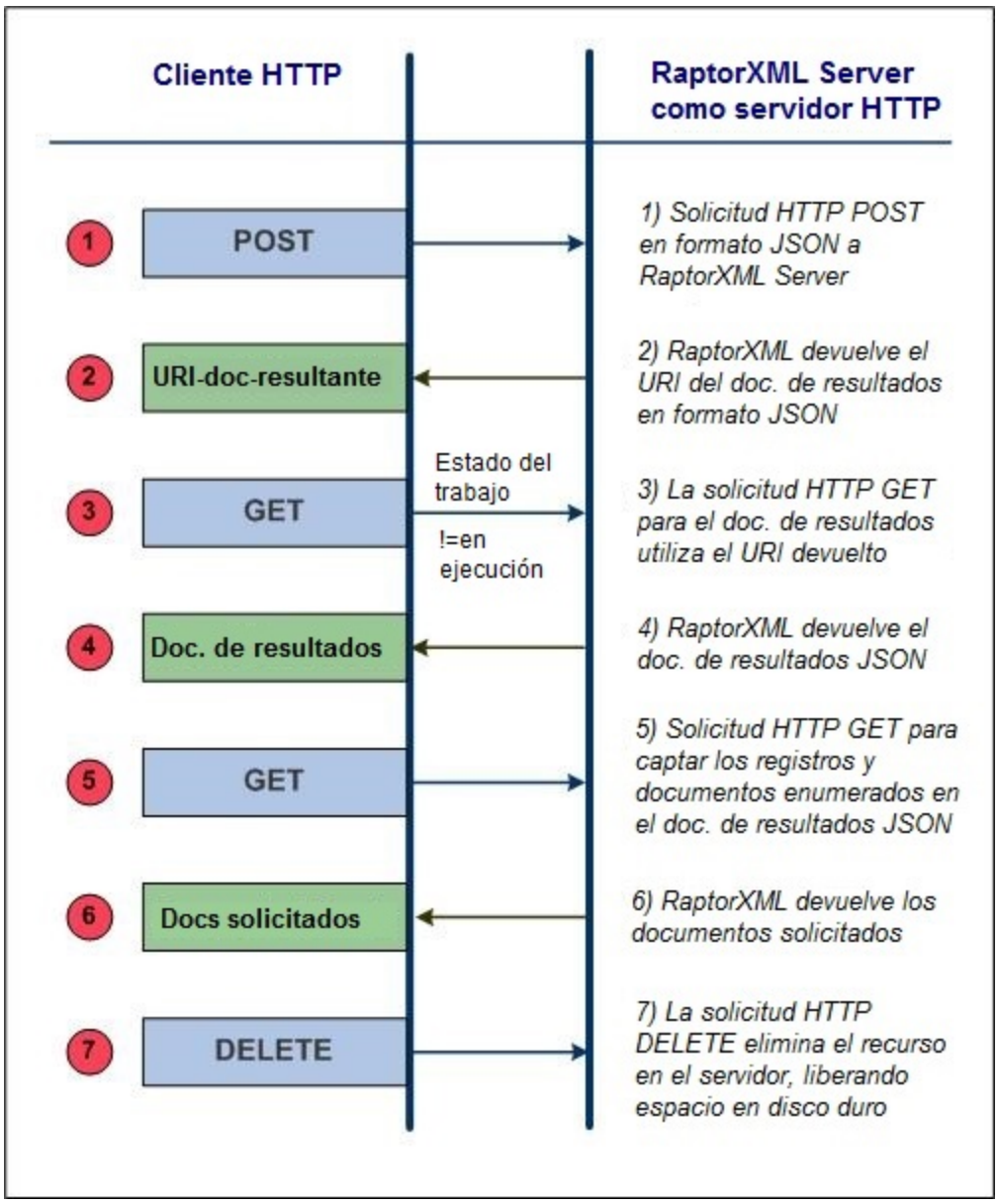
Tras iniciar RaptorXML Server como servicio, podrá acceder a sus funciones cualquier cliente HTTP que pueda:

- 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.

© 2013 Altova GmbH



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 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. 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). Dicha solicitud recibe el nombre de **trabajo de RaptorXML Server**.

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. 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 (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. 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 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 en respuesta a las solicitudes `GET` realizadas en el paso nº5.

7. El cliente puede eliminar del servidor documentos no deseados 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.

 © 2013 Altova GmbH

4.2.1 Iniciar trabajos con POST

Temas de este apartado:

- Enviar la solicitud
- Sintaxis JSON para solicitudes POST
- Cargar archivos con la solicitud POST

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.
- 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.
- 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 . 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.

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.

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.
- 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.
- 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.

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=---MiFrontera
```

El objetivo de la frontera es delimitar las diferentes partes `form-data` del cuerpo de la solicitud (*ver más abajo*).

 © 2013 Altova GmbH

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.

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 `---FronteraDeParte` (*ver más arriba*).

```
---FronteraDeParte                                                    1
Content-Disposition: form-data; name="msg"
Content-Type: application/json

{"command": "xsi", "options": {}, "args": []}                        2

---FronteraDeParte                                                    3
Content-Disposition: form-data; name="args"
Content-Type: multipart/mixed; boundary=---FronteraDeArgs

---FronteraDeArgs                                                     4
  Content-Disposition: attachment; filename="Primero.xml"
  Content-Type: application/octet-stream

  <?xml version="1.0" encoding="UTF-8"?>                              5
  <test xsi:noNamespaceSchemaLocation="Demo.xsd" xmlns:xsi="
  http://www.w3.org/2001/XMLSchema-instance">42</test>

---FronteraDeArgs                                                     6
  Content-Disposition: attachment; filename="Segundo.xml"
  Content-Type: application/octet-stream

  <?xml version="1.0" encoding="UTF-8"?>                              7
  <test xsi:noNamespaceSchemaLocation="Demo.xsd" xmlns:xsi="
  http://www.w3.org/2001/XMLSchema-instance">35</test>

---FronteraDeArgs--                                                   8
```

```
---FronteraDeParte                                                    9
Content-Disposition: form-data; name="additional_files"
Content-Type: multipart/mixed; boundary=---FronteraArchivosExtra

---FronteraArchivosExtra                                              10
  Content-Disposition: attachment; filename="Demo.xsd"
  Content-Type: application/octet-stream

  <?xml version="1.0" encoding="UTF-8"?>                              11
  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified" attributeFormDefault="unqualified">
     <xs:element name="test" type="xs:int"/>
  </xs:schema>

---FronteraArchivosExtra--                                            12
---FronteraDeParte--
```

1 El nombre de las fronteras de las principales partes `form-data` se declaran en el encabezado de la solicitud. 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 `POST`. Si `args` contiene una referencia a un archivo y se cargaron más archivos, ambos grupos de archivos se pasan al servidor.

3 La frontera indica el final de la parte `form-data` **`msg`** anterior y el inicio de la nueva parte `form-data` **`args`**. Esta parte puede tener otras partes anidadas y demarcadas por un nombre de frontera. En este ejemplo se trata de `FronteraDeArgs`.

4 El primer miembro de la matriz `args` es un archivo adjunto llamado `Primero.xml`.

5 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`**).

6 El segundo miembro de la matriz **`args`** es un archivo adjunto llamado `Segundo.xml`. Se separa del primer miembro de la matriz con la cadena de frontera `FronteraDeArgs` .

7 El texto del archivo `Segundo.xml`. También contiene una referencia al esquema `Demo.xsd` (*ver el punto 10*).

8 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`**.

9 La frontera de la parte `form-data` indica el final de la parte de `form-data` **`args`** y el inicio de la parte `form-data` **`additional_files`**. Esta última tiene una cadena de frontera diferente a las anteriores.

10 La parte **`additional_files`** contiene los metadatos del archivo adjunto `Demo.xsd`.

11 El texto del archivo `Demo.xsd`.

12 El final de la parte **`additional_files`**.

 © 2013 Altova GmbH

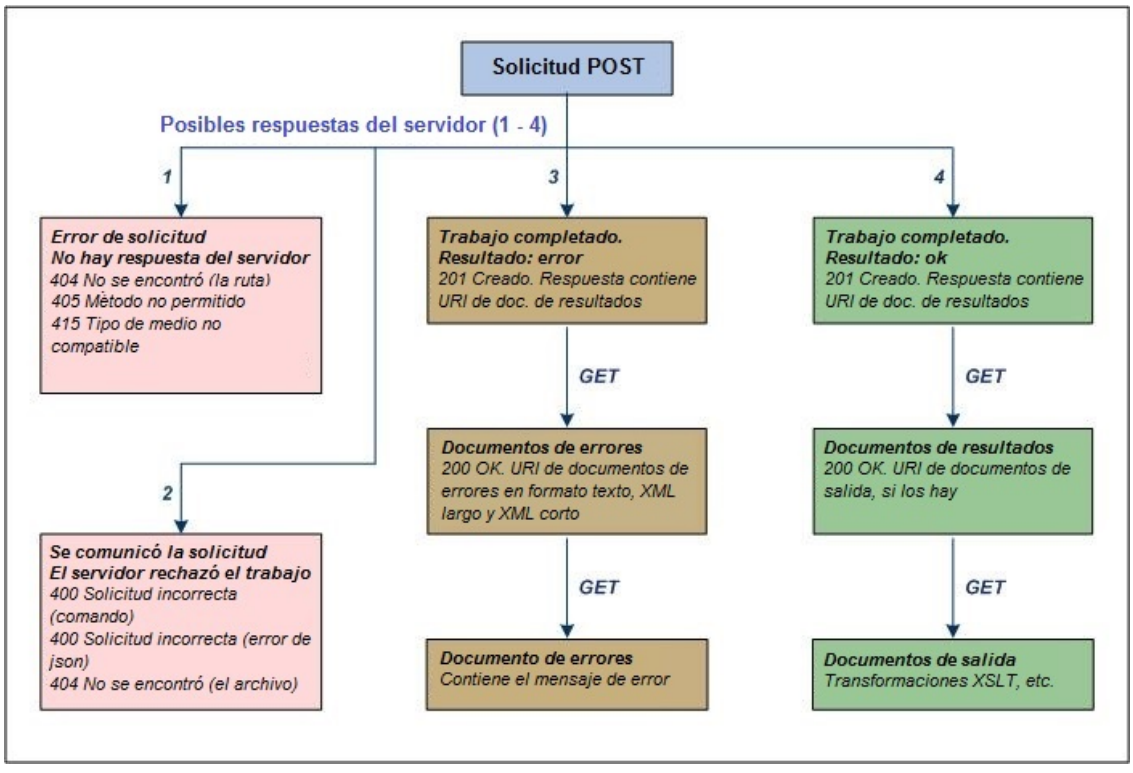
4.2.2 Respuesta del servidor a solicitudes POST

Temas de este apartado:

- Posibles respuestas del servidor
- Respuesta: Error de solicitud, sin respuesta del servidor
- Respuesta: Se comunicó la solicitud, pero el servidor rechazó el trabajo
- Respuesta: Se ejecutó el trabajo (con resultados positivos o negativos)

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, 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:



Las posibles respuestas a la solicitud `POST` del cliente son:

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
400 Solicitud incorrecta (*comando malo*)	El comando de RaptorXML no es correcto.
400 Solicitud incorrecta (*error de JSON*)	Hay un error de sintaxis JSON en el cuerpo de la solicitud.
404 No se encontró (*el archivo*)	Revise la sintaxis del URI (o ruta) de archivo 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.
- *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).

 © 2013 Altova GmbH

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 y Obtener documentos de errores y de salida.

4.2.3 Obtener el documento de resultados

Temas de este apartado:

- El URI del documento de resultados
- Recuperar el documento de resultados
  - Documento de resultados con los URI de los documentos de errores
  - Documento de resultados con los URI de los documentos de salida
  - Documento de resultados sin URI
- Acceder a los documentos de salida y de errores enumerados en el documento de resultados

---

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. Por ejemplo, si la dirección del servidor es `http://localhost:8087/` (la dirección según la configuración inicial), 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*) 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

 © 2013 Altova GmbH

restringido. 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). 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",
  "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*).

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/test.html"
        ]
      },
      "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*).
- 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.

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.

4.2.4 Obtener los documentos de salida/errores

Un documento de resultados puede contener los URI de archivo o los URI relativos de los documentos de errores, documentos de mensajes (como los registros) y documentos de salida . (En algunos casos el documento de resultados no contiene ningún URI). Los diferentes tipos de URI se describen más abajo.

Para acceder a estos documentos por HTTP:

1. Amplíe el URI relativo del archivo del documento de resultados hasta su URI absoluto.
2. Use el URI ampliado en una solicitud HTTP `GET` 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 al cliente (configuración predeterminada).
- En el directorio de trabajo del servidor. El servidor se configura 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` 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*)

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*). 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 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)

y el URI relativo de un archivo de errores del documento de resultados 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 y Obtener el documento de resultados.

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.

4.2.5 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 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).

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*] + [IDtrabajo]

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.

 © 2013 Altova GmbH

Altova RaptorXML 2013

Interfaz Python

5 Interfaz Python

La interfaz Python de RaptorXML Server permite acceder y recuperar datos de documentos XML y esquemas XML a través de diferentes API de Python (para XML y XSD). Los datos de los documentos de origen que se deben procesar y el método de procesamiento se especifican en un script Python que se envía a RaptorXML Server.

Las 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.

Las API de Python se describen en estos apartados:

- API de Python para XML
- API de Python para XSD

Scripts Python
Los scripts Python creados por el usuario se pueden enviar a RaptorXML con el parámetro `--script` de estos comandos:

- `valxml-withxsd (xsi)`
- `valxsd (xsd)`

Estos comandos que invocan a scripts Python se pueden usar tanto en la interfaz de la línea de comandos (ILC) como en la interfaz HTTP. El uso de scripts Python dentro de las API de Python de RaptorXML Server se documenta en los apartados Crear scripts Python y Ejecutar scripts Python.

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. 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` del archivo de configuración del servidor 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 (que es un subdirectorio de `output-root-directory`), 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 para evitar problemas de seguridad.

 © 2013 Altova GmbH

5.1 Crear scripts Python

Temas de este apartado:

- Versión de Python
- Trabajar con scripts seguros
- Pasar un script Python a RaptorXML
- Funciones Python
- Estructura simplificada del script Python
- La función Python de punto de entrada al detalle

Versión de Python
Los scripts Python deben crearse conforme a la **versión Python 3.3.1** porque esta es la versión utilizada en las API de Python.

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. 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` del archivo de configuración del servidor 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 (que es un subdirectorio de `output-root-directory`), 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 para evitar problemas de seguridad.

Pasar un script Python a RaptorXML
El script Python se pasa a RaptorXML con el parámetro `--script` de estos comandos:

- `valxml-withxsd (xsi)`
- `valxsd (xsd)`

Estos comandos se pueden usar en la interfaz de la línea de comandos o en la interfaz HTTP. Para ver algunos ejemplos consulte el apartado Ejecutar scripts Python.

Funciones Python
Los comandos que permiten acceder a la interfaz Python (*ver tabla más abajo*) son los comandos de validación y el script Python se ejecuta solamente si los archivos suministrados

con el comando son válidos. Una vez finalizada la validación, RaptorXML Server llama a una función determinada, dependiendo del comando ejecutado. La función a la que se llama (*ver tabla*) debe definirse en el script Python. La función debe definirse con dos parámetros: el primero es el objeto del trabajo y el segundo depende del comando ejecutado (*ver tabla*).

Comando	Función invocada por RaptorXML Server
valxml-withxsd (xsi)	**on_xsi_valid**(job,instance)
valxsd (xsd)	**on_xsd_valid**(job,schema)

Estructura simplificada del script Python
A continuación mostramos la estructura general de un script Python utilizada para acceder a la interfaz Python. Observe cómo se define la función Python de punto de entrada.

```
1   import os
    from altova import xml, xsd

2   def on_xsi_valid(job,instance):
      filename = os.path.join(job.output_dir,'script_out.txt')
      job.append_output_filename(filename)
      f = open(filename,'w')
      f.write(str(type(job))+'\n')
      f.write(str(job)+'\n')
      f.write(job.output_dir+'\n')
      f.close()
      filename2 = os.path.join(job.output_dir,'script_out2.txt')
      job.append_output_filename(filename2)
      f2 = open(filename2,'w')
      print_instance(f2,instance)
      f2.close()

3   CodeBlock-1
       ...
    CodeBlock-N
```

Descripción de la estructura del script:

1. Importa el módulo Python integrado `os` y después los módulos **`xml`**, **`xsd`** de la biblioteca **`altova`**.
2. La **función Python de punto de entrada** (*ver más abajo*). Puede ser una de estas cuatro:
   `on_xsi_valid``(job,instance)`, **`on_xsd_valid`**`(job,schema)`, **`on_dts_valid`**`(job,dts)`, **`on_xbrl_valid`**`(job,instance)`.
3. Bloques de código adicionales, cada uno con definiciones de función u otro código.

La función Python de punto de entrada al detalle
A continuación mencionamos aspectos importantes de la función Python de punto de entrada y

 © 2013 Altova GmbH

para ello usamos este ejemplo de definición de función:

```
def on_xsi_valid(job,instance):
  filename = os.path.join(job.output_dir,'script_out.txt')
  job.append_output_filename(filename)
  f = open(filename,'w')
  f.write(str(type(job))+'\n')
  f.write(str(job)+'\n')
  f.write(job.output_dir+'\n')
  f.close()
  filename2 = os.path.join(job.output_dir,'script_out2.txt')
  job.append_output_filename(filename2)
  f2 = open(filename2,'w')
  print_instance(f2,instance)
  f2.close
```

- La línea `def on_xsi_valid(job,instance):` inicia el bloque de definición de la función.
- La función se llama **`on_xsi_valid`**`(job,instance)` y toma dos argumentos: `job` y `instance`.
- Esta es la función que se invoca después de que RaptorXML Server ejecute el comando `valxml-withxsd (xsi)` y determine que los archivos XML suministrados son válidos.
- El valor de los argumentos `job` y `instance` vienen dados por RaptorXML Server.
- El valor de la variable `filename` se construye usando `job.output_dir`, cuyo valor, en el caso de uso con HTTP, se especifica en el archivo de configuración del servidor y, en caso de uso con la ILC, es el directorio de trabajo.
- La función `job.append_output_filename` anexa un nombre de archivo al resultado del trabajo.

5.2 Ejecutar scripts Python

Para pasar scripts Python a RaptorXML Server es necesario suministrar la URL del script como valor de la opción `--script`. La opción `--script` es compatible con estos comandos:

- `valxml-withxsd (xsi)`
- `valxsd (xsd)`

Estos comandos se pueden usar tanto en la interfaz de la línea de comandos como en la interfaz HTTP.

Ejemplos
Aquí puede ver algunos ejemplos de uso de estos comandos:

- **raptorxml** `xsi --script=xml.py --streaming=false c:\TieneRefXSD.xml`
- **raptorxml** `xsd --script=xsd.py c:\Prueba.xsd`

Nota: cuando use la opción `--script` con el comando `valxml-withxsd`, asegúrese de especificar `--streaming=false`. De lo contrario se genera una advertencia diciendo que el script no se ejecutó.

Iniciar el script
Una vez suministrado el comando y validados los archivos, RaptorXML Server llama a la función Python de punto de entrada correspondiente al comando que acaba de ejecutar y le suministra los valores de los dos argumentos de la función. Si la función de punto de entrada se define en el script con el parámetro `--script`, entonces se inicia la ejecución del script.

© 2013 Altova GmbH

5.3 Ejemplo de script 1: procesar XML

Este script Python procesa los datos del archivo **NanonullOrg.xml** (situado en la carpeta examples de la carpeta de aplicación de RaptorXML) y crea un documento de salida llamado **summary.html**. El archivo de salida contiene una tabla que muestra el número total de acciones que poseen los empleados de cada departamento.

El script se pasa en la ILC con este comando:

```
raptorxml xsi --streaming=false --script=sharesummary.py NanonullOrg.xml
```

En este apartado puede encontrar:

- El script Python con anotaciones
- El documento de salida generado por el script

5.3.1 Script

El script Python con anotaciones que aparece más abajo procesa los datos del archivo **NanonullOrg.xml** (situado en la carpeta `examples` de la carpeta de aplicación de RaptorXML) y crea un documento de salida llamado **summary.html**. Este archivo de salida contiene una tabla que muestra el número total de acciones que poseen los empleados de cada departamento.

El script se puede pasar a RaptorXML con este comando:

```
raptorxml xsi --streaming=false --script=sharesummary.py NanonullOrg.xml
```

Nota: cuando use la opción `--script` con el comando `valxml-withxsd`, asegúrese de especificar `--streaming=false`. De lo contrario se genera una advertencia diciendo que el script no se ejecutó.

Nombre de archivo: sharesummary.py

```
import os
from altova import xml

def getElemTextValue(elem):
    """Devuelve el contenido de texto de un elemento XML"""

    text = ''
    for child in elem.children:
        if isinstance(child,xml.Character):
            text += child.character_code
    return text


def getChildElemsWithName(elemParent,name):
    """Devuelve una lista con todos los elementos secundarios que tienen este
    nombre"""

    elems = []
    for child in elemParent.children:
        if isinstance(child,xml.Element) and child.local_name == name:
            elems.append(child)
    return elems


def getDepartmentName(elemDepartment):
    """Devuelve el nombre del departamento indicado en el elemento <Name>"""

    return getElemTextValue(getChildElemsWithName(elemDepartment,'Name')[0])


def getDepartmentTotalShares(elemDepartment):
    """Devuelve el número de acciones que posee cada miembro del
    departamento"""

    # Inicializar total de acciones en 0
    totalShares = 0
    # Sumar las acciones de cada <Person> del departamento
    for elemPerson in getChildElemsWithName(elemDepartment,'Person'):
        elemShares = getChildElemsWithName(elemPerson,'Shares')
        # El elemento <Shares> es opcional. Por eso es necesario comprobar si
        existe
```

```
        if len(elemShares):
            # Obtener el valor del elemento <Shares>, convertirlo en un entero y
            añadirlo a la suma total
            totalShares += int(getElemTextValue(elemShares[0]))
    return totalShares


def calcSharesPerDepartment(instance):
    """Devuelve un mapa con el número de acciones que poseen los miembros del
    departamento"""

    # Obtener el elemento XML raíz
    elemOrgChart  = instance.document.document_element
    # Comprobar si el elemento raíz es <OrgChart>
    if not elemOrgChart or elemOrgChart.local_name != 'OrgChart' or
    elemOrgChart.namespace_name != 'http://www.xmlspy.com/schemas/orgchart':
        # Si no lo es, emitir un error
        raise Error('This script must be used with a valid OrgChart instance!')

    mapSharesPerDepartment = {}
    # Recorrer cada elemento <Department> de cada <Office> y establecer el
    número de acciones que poseen los miembros del departamento
     for elemOffice in getChildElemsWithName(elemOrgChart,'Office'):
        for elemDepartment in getChildElemsWithName(elemOffice,'Department'):
            mapSharesPerDepartment[getDepartmentName(elemDepartment)] =
            getDepartmentTotalShares(elemDepartment)
    return mapSharesPerDepartment


def writeSummary(mapSharesPerDepartment,filename):
    """Escribe un resumen en el nombre de archivo indicado con el número de
    acciones para cada departamento"""

    # Abrir el archivo para escribir
    f = open(filename,'w')
    f.write('<html><title>Summary</title><body><table border="1">\n')
    f.write('<tr><th>Department</th><th>Shares</th></tr>\n')
    # Generar una fila de tabla por cada departamento con el nombre del
    departamento y su número de acciones
    for name,shares in sorted(mapSharesPerDepartment.items()):
        f.write('<tr><td>%s</td><td>%d</td></tr>\n'%(name,shares))
    f.write('</table></body></html>\n')
    # Cerrar el archivo
    f.close()


def on_xsi_valid(job,instance):
    """Este método será invocado automáticamente por RaptorXML una vez
    finalizada la validación de la instancia XML"""

    # Crear el archivo 'summary.html' en el directorio de salida del trabajo
    (cuando se ejecute desde la ILC, crear en el directorio de trabajo actual)
    filename = os.path.join(job.output_dir,'summary.html')
    # Calcular el número de acciones por departamento y escribir un resumen en
    'summary.html'
    writeSummary(calcSharesPerDepartment(instance),filename)
    # Registrar el archivo de salida 'summary.html' recién generado
    job.append_output_filename(filename)
```

5.3.2 Documento de resultados

A continuación aparece un fragmento del documento de salida `summary.html` generado por el script Python `sharesummary.py`.

Nombre de archivo: summary.html

```
<html><title>Summary</title><body>
  <table border="1">
    <tr><th>Department</th><th>Shares</th></tr>
    <tr><td>Administration</td><td>2500</td></tr>
    <tr><td>Engineering</td><td>5500</td></tr>
    <tr><td>IT & Technical Support</td><td>1750</td></tr>
    <tr><td>Marketing</td><td>3000</td></tr>
    <tr><td>Research & Development</td><td>5500</td></tr>
  </table>
</body></html>
```


© 2013 Altova GmbH

5.4 Ejemplo de script 2: cambiar el formato XML

El script Python de este ejemplo cambia el formato del archivo XML **NanonullOrg.xml** (situado en la carpeta `examples` de la carpeta de aplicación de RaptorXML). A cada elemento se le aplica una sangría con tabulaciones y cada atributo se coloca en una línea nueva (lo cual facilitará la comparación visual cuando se utilice un programa de diferenciación). El documento de salida se llama **output.xml**.

El script se pasa en la ILC con este comando:

```
raptorxml xsi --streaming=false --script=reformat.py NanonullOrg.xml
```

En este apartado puede encontrar:

- El script Python con anotaciones
- El documento de salida generado por el script

5.4.1 Script

El script Python con anotaciones que aparece a continuación (`reformat.py`) cambia el formato del archivo XML **`NanonullOrg.xml`** (situado en la carpeta `examples` de la carpeta de aplicación de RaptorXML). A cada elemento se le añade sangría con tabulaciones y cada atributo se coloca en una línea nueva (lo cual facilita la comparación visual en programas de diferenciación de datos). El documento de salida se llama **`output.xml`**.

El script se pasa en la ILC con este comando:

```
raptorxml xsi --streaming=false --script=reformat.py NanonullOrg.xml
```

Nota: cuando use la opción `--script` con el comando `valxml-withxsd`, asegúrese de especificar `--streaming=false`. De lo contrario se genera una advertencia diciendo que el script no se ejecutó.

Nombre de archivo: reformat.py

```
import os
from altova import xml, xsd

def writeCharacter(f,char,depth):
    """Generar XML para el nodo de caracteres"""

    # Ignorar los nodos de texto que contienen solo caracteres de espacios en
    blanco
    if not char.element_content_whitespace:
        # Escribir el contenido de texto
        f.write("\t"*depth + char.character_code+'\n')


def writeComment(f,comment,depth):
    """Generar XML para el nodo de comentario"""

    # Escribir el comentario
    f.write("\t"*depth + '<!-- '+comment.content+'-->\n')


def writeAttribute(f,attr,depth):
    """Generar XML para el nodo de atributo (en una línea nueva)"""

    # Buscar prefijo para el espacio de nombres en el mapa de espacios de
    nombres inscope
    prefix = None
    if attr.namespace_name:
        inscope = {}
        for namespace in attr.owner_element.inscope_namespaces:
            inscope[namespace.namespace_name] = namespace.prefix
        prefix = inscope[attr.namespace_name]
        if prefix:
            prefix += ':'
    if not prefix:
        prefix = ''

    # Escribir el atributo con su valor
    f.write("\t"*depth +
    "@"+prefix+attr.local_name+"=\""+attr.normalized_value+"\"\n")
```

 © 2013 Altova GmbH

```
def writeNSAttribute(f,attr,depth):
    """Generar XML para el nodo de atributo de espacio de nombres (en una
    línea nueva)"""

    prefix = ""
    if attr.local_name != 'xmlns':
        prefix = 'xmlns:'

    # Escribir el atributo de espacio de nombres con su valor
    f.write("\t"*depth +
    "@"+prefix+attr.local_name+"=\""+attr.normalized_value+"\"\n")


def writeChildren(f,elem,depth):
    """Generar XML para todos los nodos secundarios (con sangría según el
    nivel jerárquico)"""

    # Iterar en todos los nodos secundarios
    for child in elem.children:
        if isinstance(child,xml.Element):
            writeElement(f,child,depth)
        elif isinstance(child,xml.Comment):
            writeComment(f,child,depth)
        elif isinstance(child,xml.Character):
            writeCharacter(f,child,depth)


def writeElement(f,elem,depth):
    """Generar XML para el nodo de elemento con todos sus nodos secundarios
    (con sangría según el nivel jerárquico)"""

    # Buscar prefijo para el espacio de nombres en el mapa de espacios de
    nombres inscope
    prefix = None
    if elem.namespace_name:
        inscope = {}
        for namespace in elem.inscope_namespaces:
            inscope[namespace.namespace_name] = namespace.prefix
        prefix = inscope[elem.namespace_name]
        if prefix:
            prefix += ':'
    if not prefix:
        prefix = ''

    if len(list(elem.attributes)) + len(list(elem.namespace_attributes)) == 0:
        # Escribir la etiqueta de apertura entera (sin atributos)
        f.write("\t"*depth + "<"+prefix+elem.local_name+'>\n')
    else:
        # Escribir la etiqueta de apertura sin el '>' de cierre
        f.write("\t"*depth + "<"+prefix+elem.local_name+'\n')

        # Escribir los atributos de espacio de nombres en líneas nuevas
        for attr in elem.namespace_attributes:
        writeNSAttribute(f,attr,depth+1)
        # Escribir atributos en líneas nuevas
        for attr in elem.attributes:
        writeAttribute(f,attr,depth+1)
        # Cerrar la etiqueta de apertura
        f.write("\t"*depth + ">\n")

    # Escribir todos los secundarios del elemento
    writeChildren(f,elem,depth+1)

    # Escribir la etiqueta de cierre
    f.write("\t"*depth + "</"+prefix+elem.local_name+">\n")
```

```
def writeInstance(instance,filename):
    """Generar XML para la instancia dada, donde cada elemento tiene sangría
    de tabulaciones y cada atributo aparece en una línea nueva"""

    # Abrir archivo de salida
    f = open(filename,'w')
    # Escribir el contenido de la instancia XML
    writeChildren(f,instance.document,0)
    # Cerrar el archivo de salida
    f.close()


def on_xsi_valid(job,instance):
    """Este método será invocado automáticamente por RaptorXML una vez
    finalizada la validación de la instancia XML"""

    # Crear el archivo 'output.xml' en el directorio de salida del trabajo
    (cuando se ejecute desde la ILC, crear en el directorio de trabajo actual)
    filename = os.path.join(job.output_dir,'output.xml')
    # Escribir una versión del XML de instancia con otro formato: cada
    atributo aparece en una línea nueva
    writeInstance(instance,filename)
    # Registrar el archivo de salida 'output.html' recién generado
    job.append_output_filename(filename)
```

 © 2013 Altova GmbH

5.4.2 Documento de resultados

A continuación aparece un fragmento del documento de salida `output.html` generado por el script Python `reformat.py`.

Nombre de archivo: output.xml

```
<OrgChart
	@xmlns="http://www.xmlspy.com/schemas/orgchart"
	@xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	@xmlns:ipo2="http://www.altova.com/IPO"
	@xmlns:ts="http://www.xmlspy.com/schemas/textstate"
	@xsi:schemaLocation="http://www.xmlspy.com/schemas/orgchart
OrgChart.xsd"
>
	<CompanyLogo
		@href="http://www.altova.com/nanonull.gif"
	>
	</CompanyLogo>
	<Name>
		Organization Chart
	</Name>
	<Office>
		<Name>
			Nanonull, Inc.
		</Name>
		<Desc>
			<para>
				The company was established
				<Style
					@css="font-weight: bold"
				>
					in Beverly in 1995
				</Style>
				 as a privately held software company. Since 1996,
Nanonull has been actively involved in developing nanoelectronic software
technologies. It released the first version of its acclaimed
				<Style
					@css="font-style: italic"
				>
					NanoSoft Development Suite
				</Style>
				 in February 1999. Also in 1999, Nanonull increased
its capital base with investment from a consortium of private investment
firms. The company has been expanding rapidly ever since.
			</para>
			<para>
				Due to the fact that nanoelectronic software
components are new and that sales are restricted to corporate customers,
Nanonull and its product line have not received much media publicity in the
company's early years. This has however changed in recent months as trade
journals have realized the importance of this revolutionary technology.
			</para>
		</Desc>
		<Location>
			US
		</Location>
		<Address
			@xsi:type="ipo2:US-Address"
		>
			<ipo2:street
				@xmlns:ipo="http://www.altova.com/IPO"
			>
```

```
                900 Cummings Center
        </ipo2:street>
        <ipo2:city>
                Boston
        </ipo2:city>
        <ipo2:state>
                MA
        </ipo2:state>
        <ipo2:zip>
                3234
        </ipo2:zip>
</Address>
<Phone>
        +1 (321) 555 5155 0
</Phone>
<Fax>
        +1 (321) 555 5155 4
</Fax>
<EMail>
        office@nanonull.com
</EMail>
<Department>
        <Name>
                Administration
        </Name>
        <Person
                @union="fred"
        >
                <First>
                        Vernon
                </First>
                <Last>
                        Callaby
                </Last>
                <Title>
                        Office Manager
                </Title>
                <PhoneExt>
                        582
                </PhoneExt>
                <EMail>
                        v.callaby@nanonull.com
                </EMail>
                <Shares>
                        1500
                </Shares>
                <LeaveTotal>
                        25
                </LeaveTotal>
                <LeaveUsed>
                        4
                </LeaveUsed>
                <LeaveLeft>
                        21
                </LeaveLeft>
                <union>
                        3
                </union>
                <list>
                        abc def
                </list>
                <bool>
                        true
                </bool>
                <idref>
                        fred
```

 © 2013 Altova GmbH

```
                                    </idref>
                                    <idrefs>
                                          fred joe
                                    </idrefs>
                                    <entity>
                                          myUnparsedEntity
                                    </entity>
                                    <notation>
                                          Altova-Orgchart
                                    </notation>
                              </Person>
                               ...
                        </Department>
                         ...
                  </Office>
                  ...
            </OrgChart>
```

5.5 API de Python: el objeto Job

clase **`Job`**
Una clase `Job` representa un trabajo de validación de RaptorXML.

La clase `Job` ofrece este atributo de instancia (solo lectura):

`Job.output_dir`
Devuelve la URL de un archivo con el directorio de salida del trabajo (en modo servidor) o, si no, el directorio de trabajo actual.

La clase `Job` ofrece este **método de instancia**:

`Job.append_ouptput_filename(filename)`
Añade un nombre de archivo de salida adicional a la lista de archivos de salida del trabajo. Esta lista aparece en el resultado de la línea de comadnos y en el documento de resultados del trabajo que se devuelve en la interfaz HTTP.

© 2013 Altova GmbH

5.6 API de Python para XML

El modulo **`xml`** ofrece una interfaz Python para la especificación XML Infoset, que utiliza la implementación C++ subyacente de Infoset. La interfaz Python permite al usuario navegar por el árbol del documento XML y acceder a información desde cualquier nodo XML.

Tipos disponibles
Los tipos disponibles de la API de Python para XML aparecen a continuación y se describen en los apartados de esta sección.

clase xml.Document
La clase `Document` representa un documento XML y expone las propiedades del elemento información `document` definido en la especificación XML Infoset.

clase xml.Element
La clase `Element` representa un elemento XML y expone las propiedades del elemento información `element` definido en la especificación XML Infoset.

clase xml.Attribute
La clase `Attribute` representa un atributo XML y expone las propiedades del elemento información `attribute` definido en la especificación XML Infoset.

clase xml.NSAttribute
La clase `NSAttribute` representa un espacio de nombres XML y expone las propiedades del elemento información `attribute` definido en la especificación XML Infoset.

clase xml.ProcessingInstruction
La clase `ProcessingInstruction` representa una instrucción de procesamiento XML y expone las propiedades del elemento información `processing instruction` definido en la especificación XML Infoset.

clase xml.UnexpandedEntityReference
La clase `UnexpandedEntityReference` representa una referencia de entidad XML sin expandir y expone las propiedades del elemento información `unexpanded entity reference` definido en la especificación XML Infoset.

clase xml.Character
La clase `Character` representa datos de caracteres XML y expone las propiedades del elemento información `character` definido en la especificación XML Infoset.

clase xml.Comment
La clase `Comment` representa un comentario XML y expone las propiedades del elemento información `comment` definido en la especificación XML Infoset.

clase xml.UnparsedEntity
La clase `UnparsedEntity` representa una entidad XML sin analizar y expone las propiedades del elemento información `unparsed entity` definido en la especificación XML Infoset.

clase xml.Notation
La clase `Notation` representa una notación XML y expone las propiedades del elemento información de `notación` definido en la especificación XML Infoset.

clase xml.Namespace

La clase `Namespace` representa un enlace de espacio de nombres XML y expone las propiedades del elemento información `namespace` definido en la especificación XML Infoset.

clase xml.QName

La clase `QName` representa un nombre XML completo.

© 2013 Altova GmbH

5.6.1 xml.Document

clase **xml.Document**
Un objeto `Document` representa un elemento de información de documento XML y ofrece estos **atributos de instancia** (solo lectura):

Document.children
Una lista ordenada de los elementos de información secundarios, por orden de documento. La lista contiene exactamente un elemento de información `element`. La lista también incluye un elemento de información `processing instruction` por cada instrucción de procesamiento situada fuera del elemento `document` y un elemento de información `comment` por cada comentario situado fuera del elemento `document`. Las instrucciones de procesamiento y los comentarios situados dentro de la DTD se excluyen. Si hay una declaración de tipo de documento, la lista también incluye un elemento de información de declaración de tipo de documento.

Document.document_element
El elemento de información `element` correspondiente al elemento de documento.

Document.notations
Un conjunto sin ordenar de elementos de información `notation`, uno por cada notación declarada en la DTD.

Document.unparsed_entities
Un conjunto sin ordenar de elementos de información `unparsed entity`, uno por cada entidad sin analizar declarada en la DTD.

Document.base_URI
El URI base de la entidad de documento.

Document.character_encoding_scheme
El nombre de la combinación de codificación de caracteres en la que se expresa la entidad de documento.

Document.standalone
Una indicación del estado independiente del documento: `True` o `False`. Este atributo se deriva de la declaración de documento independiente opcional del principio de la entidad de documento y devuelve el valor `None` si no hay declaración de documento independiente.

Document.version
Una cadena que representa la versión XML del documento. Este atributo se deriva de la declaración XML presente de forma opcional al principio de la entidad de documento. El valor es `None` si no existe la declaración XML.

5.6.2 xml.Element

clase **`xml.Element`**
Un objeto `Element` representa un elemento de información de elemento XML y ofrece estos **atributos de instancia** (solo lectura):

`Element.namespace_name`
El nombre de espacio de nombres, si existe, del tipo de elemento. Si el elemento no pertenece a un espacio de nombres, este atributo tiene el valor `None`.

`Element.local_name`
La parte local del nombre de tipo de elemento. No incluye ningún prefijo de espacio de nombres ni los dos puntos.

`Element.prefix`
La parte de prefijo de espacio de nombres del nombre de tipo de elemento. Si el nombre no tiene prefijo, este atributo tiene el valor `None`.

`Element.children`
Una lista ordenada de los elementos de información secundarios, por orden de documento. Esta lista contiene elementos de información `element`, `processing instruction`, `unexpanded entity reference`, `character` y `comment`, uno por cada elemento, instrucción de procesamiento, referencia de entidad sin expandir, caracteres y comentario que aparezca dentro del elemento actual. Si el elemento está vacío, esta lista no tiene miembros.

`Element.attributes`
Un conjunto sin ordenar de elementos de información `attribute`, uno por cada atributo (especificado o predeterminado de la DTD) de este elemento. Las declaraciones de espacio de nombres no aparecen en este conjunto. Si el elemento no tiene atributos, este conjunto no tiene miembros.

`Element.namespace_attributes`
Un conjunto sin ordenar de elementos de información `attribute`, uno por cada declaración de espacio de nombres (especificada o predeterminada de la DTD) de este elemento. Las declaraciones con formato `xmlns=""` y `xmlns:name=""`, que revocan la declaración del espacio de nombres predeterminado y de los prefijos, cuentan como declaraciones de espacio de nombres. La revocación de declaración de prefijo se añadió en *Namespaces XML 1.1*. Por definición, todos los atributos de espacio de nombres (incluidos los que se llaman `xmlns`, cuyo atributo `prefix` no tiene valor) tienen un URI de espacio de nombres de `http://www.w3.org/2000/xmlns/`. Si el elemento no tiene declaraciones de espacio de nombres, este conjunto no tiene miembros.

`Element.inscope_namespaces`
Un conjunto sin ordenar de elementos de información `namespace`, uno por cada espacio de nombres para este elemento. Este conjunto incluye un elemento con el prefijo `xml`, que está enlazado de forma implícita al nombre de espacio de nombres `http://www.w3.org/XML/1998/namespace`. No contiene un elemento con el prefijo `xmlns` (utilizado para declarar espacios de nombres), porque una aplicación no puede encontrar elementos ni atributos con ese prefijo. El conjunto incluirá elementos de espacio de nombres correspondientes a todos los miembros de `namespace_attributes`, excepto las declaraciones de formato `xmlns=""` o `xmlns:name=""`, que no declaran un espacio de nombres sino que revocan la declaración de los espacios de nombres y prefijos predeterminados. Cuando se resuelven los prefijos de nombres completos, se debería usar este atributo, mejor que el atributo `namespace_attributes`.

`Element.base_URI`

El URI base del elemento.

`Element.parent`
El elemento de información `documento` o `element` que contiene este elemento de información en su atributo `children`.

5.6.3 xml.Attribute

clase **xml.Attribute**
Un objeto `Attribute` representa un elemento de información de atributo XML. Representa solamente atributos XML normales y no atributos XML de enlace de espacio de nombres. Ofrece estos **atributos de instancia** (solo lectura):

Attribute.namespace_name
El nombre de espacio de nombres, si existe, del atributo. De lo contrario, su valor es `None`.

Attribute.local_name
La parte local del nombre del atributo. Esta parte no incluye los prefijos de espacio de nombres ni los dos puntos.

Attribute.prefix
La parte de prefijo de espacio de nombres del nombre del atributo. Si el nombre no tiene prefijo, su valor es `None`.

Attribute.normalized_value
El valor del atributo normalizado.

Attribute.specified
Una marca que indica si el atributo se especificó en la etiqueta inicial de su elemento o si se tomó por defecto de la DTD.

Attribute.owner_element
El elemento de información `element` que contiene este elemento de información en su atributo `attributes`.

 © 2013 Altova GmbH

5.6.4 xml.NSAttribute

clase **xml.NSAttribute**
Un objeto `NSAttribute` solo representa elementos de información de atributo XML de enlace de espacio de nombres. Ofrece estos **atributos de instancia** (solo lectura):

NSAttribute.namespace_name
Nombre del espacio de nombres, que siempre es `http://www.w3.org/2000/xmlns/`.

NSAttribute.local_name
Parte local del nombre del atributo. Esta parte no incluye el prefijo de espacio de nombres ni los dos puntos.

NSAttribute.prefix
Parte del prefijo de espacio de nombres del nombre del atributo. Si el nombre no tiene prefijo, el valor de este atributo es `None`.

NSAttribute.normalized_value
El valor del atributo normalizado.

NSAttribute.specified
Una marca que indica si el atributo se especificó en la etiqueta inicial de su elemento o si se tomó por defecto de la DTD.

NSAttribute.owner_element
El elemento de información `element` que contiene este elemento de información en su atributo `attributes`.

5.6.5 xml.ProcessingInstruction

clase **`xml.ProcessingInstruction`**
Un objeto `ProcessingInstruction` representa un elemento de información de instrucción de procesamiento XML. Ofrece estos **atributos de instancia** (solo lectura):

`ProcessingInstruction.target`
Una cadena de texto que representa la parte de destino de la instrucción de procesamiento.

`ProcessingInstruction.content`
Una cadena de texto que representa el contenido de la instrucción de procesamiento, excluido el destino y los espacios en blancos posteriores. Si no existe dicho contenido, el valor de este atributo será una cadena vacía.

`ProcessingInstruction.parent`
El elemento de información `document`, `element` o `DTD` que contiene este elemento de información en su atributo `children`.

© 2013 Altova GmbH

5.6.6 xml.UnexpandedEntityReference

clase **xml.UnexpandedEntityReference**
Un objeto `UnexpandedEntityReference` representa un elemento de información de referencia de entidad XML sin expandir. Ofrece estos **atributos de instancia** (solo lectura):

UnexpandedEntityReference.name
El nombre de la entidad referenciada.

UnexpandedEntityReference.parent
El elemento de información que contiene este elemento de información en su atributo `children`.

5.6.7 xml.Character

clase **xml.Character**
Un objeto `Character` representa un elemento de información de caracteres XML. Ofrece estos **atributos de instancia** (solo lectura):

Character.character_code
Una cadena que representa todos los puntos de código combinados del elemento de información de caracteres.

Character.element_content_whitespace
Valor booleano que indica si el carácter es un espacio en blanco que aparece dentro del contenido del elemento.

Character.parent
El elemento de información `element` que contiene este elemento de información en su atributo `children`.

© 2013 Altova GmbH

5.6.8 xml.Comment

clase **xml.Comment**
Un objeto `Comment` representa un elemento de información de comentario XML. Ofrece estos **atributos de instancia** (solo lectura):

Comment.content
Una cadena que representa el contenido del comentario.

Comment.parent
El elemento de información `document` o `element` que contiene este elemento de información en su atributo `children`.

5.6.9 xml.UnparsedEntity

clase **xml.UnparsedEntity**
Un objeto `UnparsedEntity` representa un elemento de información de entidad XML sin analizar y ofrece estos **atributos de instancia** (solo lectura):

UnparsedEntity.name
Nombre de la entidad.

UnparsedEntity.system_identifier
El identificador de sistema de la entidad tal y como aparece en la declaración de la entidad.

UnparsedEntity.public_identifier
Identificador público de la entidad. Si la entidad no tiene identificador público, el valor de este atributo es `None`.

UnparsedEntity.declaration_base_URI
El URI base relativo con respecto al cual se debe resolver el identificador de sistema.

UnparsedEntity.notation_name
El nombre de notación asociado con la entidad.

UnparsedEntity.notation
El elemento de información `notation` nombrado por el nombre de notación. Si la declaración no existe, el valor de este atributo es `None`.

© 2013 Altova GmbH

5.6.10 xml.Notation

clase **`xml.Notation`**
Un objeto `Notation` representa un elemento de información de notación y ofrece estos **atributos de instancia** (solo lectura):

`Notation.name`
Nombre de la notación.

`Notation.system_identifier`
El identificador de sistema de la notación tal y como aparece en la declaración de la notación. Si no se especificó, el valor de este atributo es `None`.

`Notation.public_identifier`
Identificador público de la notación. Si la notación no tiene identificador público, el valor de este atributo es `None`.

`Notation.declaration_base_URI`
El URI base relativo con respecto al cual se debe resolver el identificador de sistema.

5.6.11 xml.Namespace

clase **xml.Namespace(prefix, namespace_name)**
Un objeto `Namespace` representa un enlace de espacio de nombres XML y construye un objeto de clase `Namespace`. Son necesarios todos los argumentos. La clase `Namespace` ofrece estos atributos de instancia:

Namespace.prefix
Prefijo cuyo enlace describe este elemento. Desde el punto de vista sintáctico, es la parte del nombre de atributo seguido del prefijo `xmlns:`. Si el nombre de atributo es `xmlns` (la declaración sería del espacio de nombres predeterminado), este atributo tiene el valor `None`.

Namespace.namespace_name
Nombre de espacio de nombres al que está enlazado el prefijo.

© 2013 Altova GmbH

5.6.12 xml.QName

clase **xml.QName(local_name, namespace_name)**
Un objeto QName representa un nombre XML completo y construye un objeto de clase QName. Son necesarios todos los argumentos. La clase QName ofrece estos atributos de instancia:

QName.local_name
Parte local del nombre completo.

QName.namespace_name
Parte de espacio de nombres del nombre completo.

5.7 API de Python para XSD

El módulo **xsd** ofrece una interfaz Python para la implementación C++ de la capa del modelo de datos XML Schema. Esta interfaz Python permite al usuario navegar y acceder al documento de XML Schema y a Post Schema Validation Infoset (PSVI).

Tipos disponibles
Los tipos disponibles en la API de Python para XSD aparecen a continuación y se describen uno por uno en los apartados de esta sección.

clase xsd.Annotation
La clase `Annotation` representa anotaciones para humanos y computadores de los componentes del esquema.

clase xsd.Any
La clase `Any` permite validar elementos de información `attribute` y `element` en función de sus nombres de espacio de nombres y de sus nombres locales.

clase xsd.AnyAttribute
La clase `AnyAttribute` permite validar elementos de información `attribute` en función de su nombre de espacio de nombres y de sus nombres locales.

clase xsd.Assertion
La clase `Assertion` limita la existencia y los valores de elementos y atributos relacionados.

clase xsd.AttributeDeclaration
La clase `AttributeDeclaration` permite: (i) la validación local de los valores de elementos de información attribute usando una definición de tipo simple y (ii) especificar valores predeterminados y fijos para elementos de información `attribute`.

clase xsd.AttributeGroupDefinition
La clase `AttributeGroupDefinition` no participa en la validación propiamente dicha, pero construye definiciones de tipo complejo, completa o parcialmente. Los grupos de atributos se reconocer por su `name` y `target namespace`. Deben ser únicos en el esquema XSD.

clase xsd.AttributePSVI
La clase `AttributePSVI` contiene información PSVI sobre un atributo.

clase xsd.AttributeUse
La clase `AttributeUse` representa anotaciones para humanos y computadores de los componentes del esquema.

clase xsd.Block
La clase `Block` es parte de la definición de una declaración de elemento del esquema.

clase xsd.ComplexTypeDefinition
La clase `ComplexTypeDefinition` define las propiedades de un tipo complejo mediante los atributos de sus instancia.

clase xsd.ContentType
La clase `ContentType` especifica el tipo de contenido del elemento.

clase xsd.Defined

 © 2013 Altova GmbH

La clase `Defined` representa una palabra clave miembro del conjunto de valores permitidos para el atributo `disallowed_names` de `NamespaceConstraint`.

clase xsd.DerivationMethod
La clase `DerivationMethod` ofrece información sobre el método de derivación.

clase xsd.ENTITY
La clase `ENTITY` representa el tipo del atributo `ENTITY` del XML.

clase xsd.ElementDeclaration
La clase `ElementDeclaration` permite (i) la validación local de los valores de elementos de información `element` usando una definición de tipo, (ii) especificar valores predeterminados o fijos para elementos de información `element`, (iii) establecer la unicidad y relaciones de restricción de identidad entre los valores de elementos y atributos relacionados, (iv) controlar la sustituibilidad de elementos por medio del mecanismo de grupos de sustitución de elementos.

clase xsd.ElementPSVI
Si se comprobó la validez de un elemento de información `element`, las propiedades PSVI se devuelven en los atributos de instancia de la clase `ElementPSVI`.

clase xsd.Final
Un tipo complejo con una especificación vacía para `Final` se puede usar como definición de tipo base para otros tipos derivados por extensión o por restricción. Los valores explícitos `extension` y `restriction` evitan más derivaciones por extensión y restricción. Si se especifican todos los valores, entonces se dice que el tipo complejo es un tipo final (porque ya no son posibles más derivaciones).

clase xsd.ID
La clase `ID` representa el tipo de atributo `ID` de XML.

clase xsd.IDREF
La clase `IDREF` representa una secuencia de tipos de atributo `ID` de XML.

clase xsd.ID_IDREF_binding
La clase `ID_IDREF_binding` representa un enlace entre `ID` y `IDREF`.

clase xsd.ID_IDREF_table
La clase `ID_IDREF_table` representa un conjunto de asignaciones `ID-IDREF`.

clase xsd.IdentityConstraintDefinition
La clase `IdentityConstraintDefinition` permite la unicidad y restricciones de referencia con respecto al contenido de varios elementos y atributos.

clase xsd.Instance
La clase `Instance` representa el documento de instancia.

clase xsd.ModelGroup
La clase `ModelGroup` especifica una interpretación secuencial (`sequence`), disyuntiva (`choice`) o conjuntiva (`all`) de su atributo `particles`.

clase xsd.ModelGroupDefinition
La clase `ModelGroupDefinition` se identifica con su `name` y `target namespace`. Las identidades del modelo de grupo deben ser únicas en el esquema XSD. Las definiciones de modelos de grupo no participan en la validación pero el `term` de una `Particle` puede corresponder totalmente o en parte a un `ModelGroup` de un `ModelGroupDefinition`. El

atributo de instancia `model_group` es el `ModelGroup` para el que `ModelGroupDefinition` ofrece un nombre.

clase xsd.NCName
La clase `NCName` representa un nombre no colonizado.

clase xsd.NMTOKEN
La clase `NMTOKEN` representa el tipo de atributo `NMTOKEN` de XML.

clase xsd.NOTATION
La clase `NOTATION` representa el tipo de atributo `NOTATION` de XML.

clase xsd.Name
La clase `Name` representa un nombre XML.

clase xsd.NamespaceBinding
La clase `NamespaceBinding` permite enlazar un `namespace` a un `prefix`.

clase xsd.NamespaceConstraint
La clase `NamespaceConstraint` permite validar los elementos y atributos seleccionados con la restricción especificada.

clase xsd.NotationDeclaration
La clase `NotationDeclaration` especifica un valor de elemento o atributo válido. Las declaraciones de notación no participan en la validación propiamente dicha. Se les hace referencia dentro de cadenas de validación como miembros del tipo simple `NOTATION`. Un elemento de información `element` o `attribute` cuya definición de tipo de control o cuyo tipo de validación se deriva del tipo simple `NOTATION` solamente es válido si su valor se encuentra entre las enumeraciones de dicho tipo simple. Como consecuencia es necesario que dicho valor sea el `name` de una declaración de notación.

clase xsd.OpenContent
Un registro de la propiedad `OpenContent`. Es opcional si `variety` es solo elemento o mixto, de lo contrario debe estar ausente.

clase xsd.PSVI
La clase `PSVI` permite la evaluación de validez de elementos y atributos con esquemas.

clase xsd.Particle
La clase `Particle` contiene los componentes que contiene directa o indirectamente. Contiene directamente el componente que es el valor de su atributo `term`. Contiene indirectamente partículas, grupos, comodines y declaraciones de elemento que están en el valor de su propiedad `term`.

clase xsd.QName
La clase `QName` representa un nombre XML completo.

clase xsd.Schema
La clase `schema` contiene una colección de componentes de esquema. P. ej. definiciones de tipo y declaraciones de elemento, que tienen un espacio de nombres de destino común.

clase xsd.Scope
La clase `Scope` representa un registro de la propiedad `Scope`. Necesario.

clase xsd.Sibling
La clase `Sibling` representa una palabra clave miembro de un conjunto de valores

 © 2013 Altova GmbH

permitidos para el atributo `disallowed_names` de `NamespaceConstraint`.

clase xsd.SimpleTypeDefinition
La clase `SimpleTypeDefinition` representa tipos simples identificados por sus atributos `name` y `target namespace`.

clase xsd.TypeAlternative
La clase `TypeAlternative` es utilizada por una `ElementDeclaration` para especificar una condición (`test`) bajo la cual se utiliza un tipo determinado (`type_definition`) como definición de tipo de control para elementos de información `element` controlados por dicha `ElementDeclaration`. Cada `ElementDeclaration` puede tener varias `TypeAlternatives` en su `TypeTable`.

clase xsd.TypeTable
La definición de tipo con la que se valida un elemento de información `element` (su definición de tipo de control) puede ser diferente a la definición de tipo declarada. La propiedad `TypeTable` de un `ElementDeclaration`, que controla la asignación de tipo condicional, y el atributo `xsi:type` de un elemento de información `element` pueden provocar que la definición de tipo de control y la definición de tipo declarada no sean iguales.

clase xsd.Unbounded
La clase `Unbounded` es un valor de cadena. Representa el valor superior de la propiedad `maxOccurs`.

clase xsd.ValueConstraint
La clase `ValueConstraint` representa una propiedad de la clase `AttributeUse`.

clase xsd.XPathExpression
Para revisar una aserción, se construye una instancia del modelo de datos XPath 2.0, en la que el elemento de información `element` que se evalúa es el nodo raíz (sin primarios) y a los elementos y atributos se les asignan tipos y valores de acuerdo con las reglas de construcción del modelo de datos XPath 2.0. Cuando se evalúa con esta instancia del modelo de datos, `test` da como resultado `true` o `false`.

Objetos integrados de tipo de datos especial
`anyAtomicType`
`anySimpleType`
`anyURI`

Objetos de tipo de datos string
`language`
`normalizedString`
`string`
`token`

Objeto de tipo de datos boolean
`boolean`

Objetos de tipo de datos numeric
`byte`
`decimal`
`double`
`float`
`int`
`integer`
`long`
`negativeInteger`
`nonNegativeInteger`

nonPositiveInteger
positiveInteger
short
unsignedByte
unsignedInt
unsignedLong
unsignedShort

Objetos de tipo de datos duration

dayTimeDuration
duration
yearMonthDuration

Objetos de tipo de datos date y time

date
dateTime
dateTimeStamp
gDay
gMonth
gYear
gYearMonth
time

Objetos de tipo de datos binary

base64Binary
hexBinary

Objetos faceta

assertionsFacet
enumerationFacet
fractionDigitsFacet
lengthFacet
maxExclusiveFacet
maxInclusiveFacet
maxLengthFacet
minExclusiveFacet
minInclusiveFacet
minLengthFacet
pattern
totalDigitsFacet

© 2013 Altova GmbH

5.7.1 xsd.Annotation

clase **xsd.Annotation**
La clase `Annotation` representa anotaciones para humanos y computadores de los componentes del esquema. *Ir a la descripción.*

La clase `Annotation` ofrece estos **atributos de instancia** (de solo lectura):

Annotation.application_information
Una secuencia de elementos de información `element` para procesamiento automático.

Annotation.user_information
Una secuencia de elementos de información `element` para consumo humano.

5.7.2 xsd.Any

clase **xsd.Any**
Una clase `Any` permite validar elementos de información `attribute` y `element` en función de sus nombres de espacio de nombres y de sus nombres locales. *Ir a la descripción.*

La clase `Any` ofrece estas **constantes**:

Any.SKIP
Sin restricciones: basta con que el elemento tenga un formato XML correcto.

Any.STRICT
Debe haber una declaración de nivel superior para el elemento o el elemento debe tener un `xsi:type` y debe ser válido.

Any.LAX
Si el elemento tiene una declaración que es única, debe ser válido con respecto a dicha declaración. Esto significa: si se puede, validar. Si no, no es necesario validar.

La clase `Any` ofrece estos **atributos de instancia** (de solo lectura):

Any.annotations
Una secuencia de componentes `Annotation`.

Any.namespace_constraint
Un registro de la propiedad de restricción de espacio de nombres. Necesario.

Any.process_contents
Controla el impacto durante la evaluación de elementos de información permitida por comodines. Toma `SKIP`, `STRICT` o `LAX`. Necesario.

 © 2013 Altova GmbH

5.7.3 xsd.AnyAttribute

clase **xsd.AnyAttribute**
Una clase `AnyAttribute` permite validar elementos de información `attribute` en función de su nombre de espacio de nombres y de sus nombres locales. *Ir a la descripción*.

La clase `AnyAttribute` ofrece estas **constantes**:

`Any.SKIP`
Sin restricciones: basta con que el elemento tenga un formato XML correcto.

`Any.STRICT`
Debe haber una declaración de nivel superior para el elemento o el elemento debe tener un `xsi:type` y debe ser válido.

`Any.LAX`
Si el elemento tiene una declaración que es única, debe ser válido con respecto a dicha declaración. Esto significa: si se puede, validar. Si no, no es necesario validar.

The `AnyAttribute` ofrece estos **atributos de instancia** (de solo lectura):

`AnyAttribute.annotations`
Una secuencia de componentes `Annotation`.

`AnyAttribute.namespace_constraint`
Un registro de la propiedad de restricción de espacio de nombres. Necesario.

`AnyAttribute.process_contents`
Controla el impacto durante la evaluación de elementos de información permitida por comodines. Toma `SKIP`, `STRICT` o `LAX`. Necesario.

5.7.4 xsd.Assertion

clase **`xsd.Assertion`**
La clase `Assertion` limita la existencia y los valores de elementos y atributos relacionados. *Ir a la descripción*.

La clase `Assertion` ofrece estos **atributos de instancia** (de solo lectura):

`Assertion.annotations`
Una secuencia de componentes `Annotation`.

`Assertion.test`
Un registro de la propiedad de expresión XPath.

 © 2013 Altova GmbH

5.7.5 xsd.AttributeDeclaration

clase **xsd.AttributeDeclaration**
Una clase `AttributeDeclaration` permite: (i) la validación local de los valores de elementos de información attribute usando una definición de tipo simple y (ii) especificar valores predeterminados y fijos para elementos de información `attribute`. *Ir a la descripción.*

La clase `AttributeDeclaration` ofrece estos **atributos de instancia** (de solo lectura):

`AttributeDeclaration.annotations`
Una secuencia de componentes `Annotation`.

`AttributeDeclaration.name`
Un valor `xs:NCName`. Necesario.

`AttributeDeclaration.target_namespace`
Un valor `xs:anyURI`. Opcional.

`AttributeDeclaration.type_definition`
Un componente definición de tipo simple. Necesario.

`AttributeDeclaration.scope`
Un registro de la propiedad `Scope`. Necesario.

`AttributeDeclaration.value_constraint`
Un registro de la propiedad `Value Constraint`. Opcional.

`AttributeDeclaration.inheritable`
Un valor `xs:boolean`. Necesario.

5.7.6 xsd.AttributeGroupDefinition

clase **xsd.AttributeGroupDefinition**

La clase `AttributeGroupDefinition` no participa en la validación propiamente dicha, pero construye definiciones de tipo complejo, completa o parcialmente. Los grupos de atributos se reconocer por su `name` y `target namespace`. Deben ser únicos en el esquema XSD. *Ir a la descripción*.

La clase `AttributeGroupDefinition` ofrece estos **atributos de instancia** (de solo lectura):

AttributeGroupDefinition.annotations
Una secuencia de componentes `Annotation`.

AttributeGroupDefinition.name
Un valor `xs:NCName`. Necesario.

AttributeGroupDefinition.target_namespace
Un valor `xs:anyURI`. Opcional.

AttributeGroupDefinition.attribute_uses
Un conjunto de componentes `Attribute Use`.

AttributeGroupDefinition.attribute_wildcard
Un componente comodín. Opcional.

 © 2013 Altova GmbH

5.7.7 xsd.AttributePSVI

clase **xsd.AttributePSVI**
La clase `AttributePSVI` contiene información PSVI sobre un atributo. *Ir a la descripción*.

La clase `AttributePSVI` ofrece estos **atributos de instancia** (de solo lectura):

AttributePSVI.validity
Uno de estos casos: si se evalúa de forma estricta y es válido localmente, entonces `valid`; si se evalúa de forma estricta y no es válido localmente, entonces `invalid`; de lo contrario `notKnown`.

AttributePSVI.validation_attempted
Uno de estos casos: si se evalúa de forma estricta, entonces `full`; de lo contrario `none`.

AttributePSVI.attribute_declaration
Un elemento con la misma forma que el componente de definición de tipo de control.

AttributePSVI.schema_normalized_value
Si el valor normalizado del atributo es válido con respecto a la definición de tipo de control, entonces el valor normalizado validado. De lo contrario está ausente.

AttributePSVI.schema_actual_value
Si el valor normalizado del esquema no está ausente, entonces el valor real correspondiente. De lo contrario ausente.

AttributePSVI.type_definition
Un elemento con la misma forma que el componente de definición de tipo de control.

AttributePSVI.type_definition_type
`simple`.

AttributePSVI.type_definition_namespace
El espacio de nombres de destino de la definición de tipo.

AttributePSVI.type_definition_anonymous
`true` si el `name` de la definición de tipo está ausente, de lo contrario `false`.

AttributePSVI.type_definition_name
El `name` de la definición de tipo, si el `name` no está ausente. Si la propiedad `name` de la definición de tipo está ausente, entonces los procesadores de esquemas pueden, si quieren, ofrecer un valor que identifique de manera unívoca esta definición de tipo entre las definiciones con el mismo espacio de nombres de destino.

AttributePSVI.member_type_definition
Un elemento con la misma forma que el tipo que valida del valor real del esquema.

AttributePSVI.member_type_definition_namespace
El espacio de nombres de destino del tipo que valida.

AttributePSVI.member_type_definition_anonymous
`true` si el `name` del tipo que valida está ausente. De lo contrario `false`.

AttributePSVI.member_type_definition_name
El `name` del tipo que valida si no está ausente.

AttributePSVI.member_type_definitions

Una secuencia de componentes de la definición de tipo simple con la misma longitud que el valor real del esquema, siendo cada uno un elemento con la misma forma que el tipo de validación del elemento correspondiente del valor real del esquema.

© 2013 Altova GmbH

5.7.8 xsd.AttributeUse

clase **xsd.AttributeUse**
La clase `AttributeUse` es un componente de utilidad que controla la aparición y el comportamiento predeterminado de las declaraciones de atributo. Desempeña el mismo papel para las declaraciones de atributos en tipos complejos que el papel que desempeñan las partículas para las declaraciones de elemento. *Ir a la descripción*

La clase `AttributeUse` ofrece estos **atributos de instancia** (solo lectura):

`AttributeUse.annotations`
Secuencia de componentes `Annotation`.

`AttributeUse.required`
Un valor `xs:boolean`. Necesario

`AttributeUse.attribute_declaration`
Un componente `AttributeDeclaration`. Necesario.

`AttributeUse.value_constraint`
Un registro de la propiedad `ValueConstraint`. Opcional.

`AttributeUse.inheritable`
Un valor `xs:boolean`. Necesario.

5.7.9 xsd.Block

clase **xsd.Block**
Parte de la definición de una declaración de elemento del esquema. Necesaria.
Ir a la descripción

La clase `Block` ofrece estas **constantes**:

```
Block.NONE
Block.EXTENSION
Block.RESTRICTION
Block.SUBSTITUTION
```


© 2013 Altova GmbH

5.7.10 xsd.ComplexTypeDefnition

clase **xsd.ComplexTypeDefinition**
Una clase `ComplexTypeDefinition` define las propiedades de un tipo complejo a través de sus atributos de instancia (*ver lista*).
Ir a la descripción.

La clase `ComplexTypeDefinition` ofrece estos **atributos de instancia** (solo lectura):

ComplexTypeDefinition.annotations
Secuencia de componentes `Annotation`.

ComplexTypeDefinition.name
Un valor `xs:NCName`. Opcional.

ComplexTypeDefinition.target_namespace
Un valor `xs:anyURI`. Opcional.

ComplexTypeDefinition.base_type_definition
Un componente de definición de tipo. Necesario.

ComplexTypeDefinition.final
Un subconjunto de `{extension, restriction}`.

ComplexTypeDefinition.context
Necesario si el atributo de instancia `name` está ausente (*ver más arriba*). De lo contrario, debe estar ausente. Bien un `ElementDeclaration` o un `ComplexTypeDefinition`.

ComplexTypeDefinition.derivation_method
Uno de `{extension, restriction}`. Necesario.

ComplexTypeDefinition.abstract
Un valor `xs:boolean`. Necesario.

ComplexTypeDefinition.attribute_uses
Un conjunto de componentes `AttributeUse`.

ComplexTypeDefinition.attribute_wildcard
Un componente comodín. Opcional.

ComplexTypeDefinition.content_type
Un registro de la propiedad `ContentType`. Necesario.

ComplexTypeDefinition.prohibited_substitutions
Un subconjunto de `{extension, restriction}`.

ComplexTypeDefinition.assertions
Una secuencia de componentes `Assertion`.

5.7.11 xsd.ContentType

clase **xsd.ContentType**
La clase `ContentType` especifica el tipo de contenido del elemento.
Ir a la descripción

La clase `ContentType` ofrece estas **constantes**:

ContentType.EMPTY
ContentType.SIMPLE
ContentType.ELEMENT_ONLY
ContentType.MIXED

La clase `ContentType` ofrece estos **atributos de instancia** (solo lectura):

ContentType.variety
Uno de estos: `{empty, simple, element-only, mixed}`. Necesario.

ContentType.particle
Un componente `Particle`. Necesario si `{variety}` es `element-only` o `mixed`, de lo contrario debe estar ausente.

ContentType.open_content
Un registro de la propiedad `OpenContent`. Opcional si `{variety}` es `element-only` o `mixed`, de lo contrario debe estar ausente.

ContentType.simple_type_definition
Un componente `SimpleTypeDefinition`. Necesario si `{variety}` es `simple`, de lo contrario debe estar ausente.

 © 2013 Altova GmbH

5.7.12 xsd.Defined

clase **xsd.Defined**

La clase `Defined` representa una palabra clave miembro del conjunto de valores permitidos para el atributo `disallowed_names` de `NamespaceConstraint`. *Ir a la descripción*

La clase `Defined` ofrece este **método de instancia**:

Defined.__str__()

5.7.13 xsd.DerivationMethod

clase **xsd.DerivationMethod**
La clase `DerivationMethod` ofrece información sobre el método de derivación.
Ir a la descripción

La clase `DerivationMethod` ofrece estas **constantes**:

```
DerivationMethod.NONE
DerivationMethod.RESTRICTION
DerivationMethod.EXTENSION
DerivationMethod.LIST
DerivationMethod.UNION
```


© 2013 Altova GmbH

5.7.14 xsd.ENTITY

clase **xsd.ENTITY**
Representa el tipo de atributo ENTITY de XML. *Ir a la descripción*

La clase ENTITY ofrece estos **atributos de instancia** (solo lectura):

ENTITY.value
Una cadena que ofrece el valor de la entidad.

5.7.15 xsd.ElementDeclaration

clase **xsd.ElementDeclaration**

La clase `ElementDeclaration` permite (i) la validación local de los valores de elementos de información `element` usando una definición de tipo, (ii) especificar valores predeterminados o fijos para elementos de información `element`, (iii) establecer la unicidad y relaciones de restricción de identidad entre los valores de elementos y atributos relacionados, (iv) controlar la sustituibilidad de elementos por medio del mecanismo de grupos de sustitución de elementos. *Ir a la descripción*

La clase `ElementDeclaration` ofrece estos **atributos de instancia** (solo lectura):

ElementDeclaration.annotations
Una secuencia de componentes `Annotation`.

ElementDeclaration.name
Un valor `xs:NCName`. Necesario.

ElementDeclaration.target_namespace
Un valor `xs:anyURI`. Opcional.

ElementDeclaration.type_definition
Un componente Tipo de definición. Necesario.

ElementDeclaration.type_table
Un registro de la propiedad `TypeTable`. Opcional.

ElementDeclaration.scope
Un registro de la propiedad `Scope`. Necesario.

ElementDeclaration.value_constraint
Un registro de la propiedad `ValueConstraint`. Opcional.

ElementDeclaration.nillable
Un valor `xs:boolean`. Necesario.

ElementDeclaration.identity_constraint_definitions
Un conjunto de componentes `IdentityConstraintDefinition`.

ElementDeclaration.substitution_group_affiliations
Un conjunto de componentes `ElementDeclaration`.

ElementDeclaration.substitution_group_exclusions
Un subconjunto de `{extension, restriction}`.

ElementDeclaration.disallowed_substitutions
Un subconjunto de `{substitution, extension, restriction}`.

ElementDeclaration.abstract
Un valor `xs:boolean`. Necesario.

 © 2013 Altova GmbH

5.7.16 xsd.ElementPSVI

clase **xsd.ElementPSVI**
Si se comprobó la validez de un elemento de información `element`, las propiedades PSVI se devuelven en los atributos de instancia de la clase `ElementPSVI`. *Ir a la descripción*

La clase `ElementPSVI` ofrece estos **atributos de instancia** (solo lectura):

ElementPSVI.validity
Uno de estos: `valid`, `invalid` o `notKNown`. *Ir a la descripción*.

ElementPSVI.validation_attempted
Uno de estos `full`, `none` o `partial`. *Ir a la descripción*.

ElementPSVI.element_declaration
Un elemento con la misma forma que el componente de declaración de control mismo.

ElementPSVI.nil
Un valor de `true` si se cumple con la cláusula 3.2.3 de Elemento válido localmente (Element) . De lo contrario, `false`.

ElementPSVI.schema_normalized_value
Si una definición de tipo de control es conocida para un elemento de información `element`, entonces en el conjunto de información posterior a la validación con esquema, el valor del elemento. Ver *Element information items*.

ElementPSVI.schema_actual_value
Si una definición de tipo de control es conocida para un elemento de información `element`, entonces en el conjunto de información posterior a la validación con esquema, el valor del elemento. Ver *Element information items*.

ElementPSVI.type_definition
Si una definición de tipo de control es conocida para un elemento de información `element`, entonces en el conjunto de información posterior a la validación con esquema, el valor del elemento. Ver *Element information items*.

ElementPSVI.type_definition_type
Si una definición de tipo de control es conocida para un elemento de información `element`, entonces en el conjunto de información posterior a la validación con esquema, el valor del elemento. Ver *Element information items*.

ElementPSVI.type_definition_namespace
Si una definición de tipo de control es conocida para un elemento de información `element`, entonces en el conjunto de información posterior a la validación con esquema, el valor del elemento. Ver *Element information items*.

ElementPSVI.type_definition_anonymous
Si una definición de tipo de control es conocida para un elemento de información `element`, entonces en el conjunto de información posterior a la validación con esquema, el valor del elemento. Ver *Element information items*.

ElementPSVI.type_definition_name
Si una definición de tipo de control es conocida para un elemento de información `element`, entonces en el conjunto de información posterior a la validación con esquema, el valor del elemento. Ver *Element information items*.

ElementPSVI.member_type_definition
Si una definición de tipo de control es conocida para un elemento de información `element`, entonces en el conjunto de información posterior a la validación con esquema, el valor del elemento. Ver *Element information items*.

ElementPSVI.member_type_definition_namespace
Si una definición de tipo de control es conocida para un elemento de información `element`, entonces en el conjunto de información posterior a la validación con esquema, el valor del elemento. Ver *Element information items*.

ElementPSVI.member_type_definition_anonymous
Si una definición de tipo de control es conocida para un elemento de información `element`, entonces en el conjunto de información posterior a la validación con esquema, el valor del elemento. Ver *Element information items*.

ElementPSVI.member_type_definition_name
Si una definición de tipo de control es conocida para un elemento de información `element`, entonces en el conjunto de información posterior a la validación con esquema, el valor del elemento. Ver *Element information items*.

ElementPSVI.member_type_definitions
Si una definición de tipo de control es conocida para un elemento de información `element`, entonces en el conjunto de información posterior a la validación con esquema, el valor del elemento. Ver *Element information items*.

ElementPSVI.inherited_attributes
Una lista de elementos de información `attribute` heredados. Las reglas de herencia se describe *aquí*.

 © 2013 Altova GmbH

5.7.17 xsd.Final

clase **xsd.Final**

Un tipo complejo con una especificación vacía para `Final` se puede usar como definición de tipo base para otros tipos derivados por extensión o por restricción. Los valores explícitos `extension` y `restriction` evitan más derivaciones por extensión y restricción. Si se especifican todos los valores, entonces se dice que el tipo complejo es un tipo final (porque ya no son posibles más derivaciones). *Ir a la descripción*

La clase `Final` ofrece estas **constantes**:

```
Final.NONE
Final.EXTENSION
Final.RESTRICTION
Final.LIST
Final.UNION
```

5.7.18 xsd.ID

clase **xsd.ID**
Representa el tipo de atributo ID de XML. *Ir a la descripción*

La clase ID ofrece estos **atributos de instancia** (solo lectura):

ID.value
Una cadena que da el valor del ID.

 © 2013 Altova GmbH

5.7.19 xsd.IDREF

clase **xsd.IDREF**
Representa una secuencia de tipos de atributo ID de XML. *Ir a la descripción*

La clase IDREF ofrece estos **atributos de instancia** (solo lectura):

IDREF.value
Una secuencia de valores ID.

5.7.20 xsd.ID_IDREF_binding

clase **xsd.ID_IDREF_binding**
La clase `ID_IDREF_binding` representa un enlace entre `ID` y `IDREF`. *Ir a la descripción*

La clase `ID_IDREF_binding` ofrece estos **atributos de instancia** (solo lectura):

ID_IDREF_binding.id
ID_IDREF_binding.binding

5.7.21 xsd.ID_IDREF_table

clase **xsd.ID_IDREF_table**
La clase `ID_IDREF_table` representa un conjunto de asignaciones `ID-IDREF`. *Ir a la descripción*

La clase `ID_IDREF_table` ofrece estos **métodos de instancia**:

```
ID_IDREF_table.__len__()
ID_IDREF_table.__iter__()
```

5.7.22 xsd.IdentityConstraintDefinition

clase **xsd.IdentityConstraintDefinition**
La clase `IdentityConstraintDefinition` permite la unicidad y restricciones de referencia con respecto al contenido de varios elementos y atributos. *Ir a la descripción*

La clase `IdentityConstraintDefinition` ofrece estas **constantes**:

IdentityConstraintDefinition.KEY
La definición de restricción de identidad confirma la singularidad igual que con `unique`. La constante `key` confirma que todo el contenido seleccionado tiene de hecho esas tuplas.

IdentityConstraintDefinition.KEYREF
La definición de restricción de identidad confirma una correspondencia con respecto al contenido identificado por `selector` o las tuplas resultantes de evaluar la expresión XPath de `field` con las de la constante `key` referenciada.

IdentityConstraintDefinition.UNIQUE
La definición de restricción de identidad confirma la singularidad con respecto al contenido identificado por `selector` o de las tuplas resultantes de evaluar las expresiones XPath de `field`.

La clase `IdentityConstraintDefinition` ofrece estos **atributos de instancia** (solo lectura):

IdentityConstraintDefinition.annotations
Una secuencia de componentes `Annotation`.

IdentityConstraintDefinition.name
Un valor `xs:NCName`. Necesario.

IdentityConstraintDefinition.target_namespace
Un valor `xs:anyURI`. Opcional.

IdentityConstraintDefinition.identity_constraint_category
Uno de estos: `{key, keyref, unique}`. Necesario.

IdentityConstraintDefinition.selector
Un registro de la propiedad `XPathExpression`. Necesario.

IdentityConstraintDefinition.fields
Una secuencia de registros de la propiedad `XPathExpression`.

IdentityConstraintDefinition.referenced_key
Un componente `IdentityConstraintDefinition`. Necesario si `identity_constraint_category` es `keyref`, de lo contrario (si `identity_constraint_category` es `key` o `unique`) debe estar ausente. Si está presente un valor, su `identity_constraint_category` debe ser `key` o `unique`.

 © 2013 Altova GmbH

5.7.23 xsd.Instance

clase **xsd.Instance**

La clase `Instance` representa el documento de instancia. *Ir a la descripción*

La clase `Instance` ofrece estos **atributos de instancia** (solo lectura):

Instance.filename
Instance.document
Instance.psvi
Instance.schema

5.7.24 xsd.ModelGroup

clase **xsd.ModelGroup**

La clase `ModelGroup` especifica una interpretación secuencial (`sequence`), disyuntiva (`choice`) o conjuntiva (`all`) de su atributo `particles`. *Ir a la descripción*

La clase `ModelGroup` ofrece estas **constantes**:

ModelGroup.ALL
Determina si los secundarios del elemento de información `element` validados por el modelo de grupo deben corresponderse con las `particles` especificadas. Los elementos pueden aparecer en cualquier orden.

ModelGroup.CHOICE
Determina si los secundarios del elemento de información `element` validados por el modelo de grupo deben corresponderse con exactamente una de las `particles` especificadas.

ModelGroup.SEQUENCE
Determina si los secundarios del elemento de información `element` validados por el modelo de grupo deben corresponderse, en el mismo orden, con las `particles` especificadas.

La clase `ModelGroup` ofrece estos **atributos de instancia** (solo lectura):

ModelGroup.annotations
Una secuencia de componentes `Annotation`.

ModelGroup.compositor
Uno de estos: `{all, choice, sequence}`. Necesario.

ModelGroup.particles
Una secuencia de componentes `Particle`.

 © 2013 Altova GmbH

5.7.25 xsd.ModelGroupDefinition

clase **xsd.ModelGroupDefinition**

La clase `ModelGroupDefinition` se identifica con su `name` y `target namespace`. Las identidades del modelo de grupo deben ser únicas en el esquema XSD. Las definiciones de modelos de grupo no participan en la validación pero el `term` de una `Particle` puede corresponder totalmente o en parte a un `ModelGroup` de un `ModelGroupDefinition`. El atributo de instancia `model_group` es el `ModelGroup` para el que `ModelGroupDefinition` ofrece un nombre. *Ir a la descripción*

La clase `ModelGroupDefinition` ofrece estos **atributos de instancia** (solo lectura):

ModelGroupDefinition.annotations
Una secuencia de componentes `Annotation`.

ModelGroupDefinition.name
Un valor `xs:NCName`. Necesario.

ModelGroupDefinition.target_namespace
Un valor `xs:anyURI`. Opcional.

ModelGroupDefinition.model_group
Un componente `ModelGroup`. Necesario.

5.7.26 xsd.NCName

clase **xsd.NCName**

La clase NCName representa un nombre no colonizado. *Ir a la descripción*

La clase NCName ofrece estos **atributos de instancia** (solo lectura):

NCName.value

 © 2013 Altova GmbH

5.7.27 xsd.NMTOKEN

clase **xsd.NMTOKEN**
La clase NMTOKEN representa el tipo de atributo NMTOKEN de XML. *Ir a la descripción*

La clase NMTOKEN ofrece estos **atributos de instancia** (solo lectura):

NMTOKEN.value

5.7.28 xsd.NOTATION

clase **xsd.NOTATION**
La clase NOTATION representa el tipo de atributo NOTATION de XML. *Ir a la descripción*

La clase NOTATION ofrece estos **atributos de instancia** (solo lectura):

NOTATION.namespace_name
NOTATION.local_part

 © 2013 Altova GmbH

5.7.29 xsd.Name

clase **xsd.Name**
La clase `Name` representa un nombre XML. *Ir a la descripción*

La clase `Name` ofrece estos **atributos de instancia** (solo lectura):

Name.value

5.7.30 xsd.NamespaceBinding

clase **xsd.NamespaceBinding**

La clase `NamespaceBinding` permite enlazar un `namespace` a un `prefix`. *Ir a la descripción*

La clase `NamespaceBinding` ofrece estos **atributos de instancia** (solo lectura):

NamespaceBinding.prefix
NamespaceBinding.namespace

© 2013 Altova GmbH

5.7.31 xsd.NamespaceConstraint

clase **xsd.NamespaceConstraint**

La clase `NamespaceConstraint` permite validar los elementos y atributos seleccionados con la restricción especificada. *Ir a la descripción*

La clase `NamespaceConstraint` ofrece estas **constantes**:

NamespaceConstraint.ANY
NamespaceConstraint.ENUMERATION
NamespaceConstraint.NOT

La clase `NamespaceConstraint` ofrece estos **atributos de instancia** (solo lectura):

NamespaceConstraint.variety
Uno de estos: `{any, enumeration, not}`. Necesario.

NamespaceConstraint.namespaces
Un conjunto, cuyos miembros son bien un valor `xs:anyURI` o bien el valor característico `absent`. Necesario.

NamespaceConstraint.disallowed_names
Un conjunto, cuyos miembros son bien un valor `xs:QName`, bien la palabra clave `defined` o bien la palabra clave `sibling`. Necesario.

5.7.32 xsd.NotationDeclaration

clase **xsd.NotationDeclaration**

La clase `NotationDeclaration` especifica un valor de elemento o atributo válido. Las declaraciones de notación no participan en la validación propiamente dicha. Se les hace referencia dentro de cadenas de validación como miembros del tipo simple `NOTATION`. Un elemento de información `element` o `attribute` cuya definición de tipo de control o cuyo tipo de validación se deriva del tipo simple `NOTATION` solamente es válido si su valor se encuentra entre las enumeraciones de dicho tipo simple. Como consecuencia es necesario que dicho valor sea el `name` de una declaración de notación. *Ir a la descripción*

La clase `NotationDeclaration` ofrece estos **atributos de instancia** (solo lectura):

NotationDeclaration.annotations
Una secuencia de componentes `Annotation`.

NotationDeclaration.name
Un valor `xs:NCName`. Necesario.

NotationDeclaration.target_namespace
Un valor `xs:anyURI`. Opcional.

NotationDeclaration.system_identifier
Un valor `xs:anyURI`. Necesario si `public_identifier` está ausente. De lo contrario, opcional.

NotationDeclaration.public_identifier
Un valor ID público. Necesario si `system_identifier` está ausente. De lo contrario, opcional.

 © 2013 Altova GmbH

5.7.33 xsd.OpenContent

clase **xsd.OpenContent**
Un registro de la propiedad `OpenContent`. Es opcional si `variety` es solo elemento o mixto, de lo contrario debe estar ausente. *Ir a la descripción*

La clase `OpenContent` ofrece estas **constantes**:

OpenContent.INTERLEAVE
OpenContent.SUFFIX

La clase `OpenContent` ofrece estos **atributos de instancia** (solo lectura):

OpenContent.mode
Uno de estos `{interleave, suffix}`. Necesario.

OpenContent.wildcard
Un componente comodín. Necesario

5.7.34 xsd.PSVI

clase **xsd.PSVI**

La clase PSVI permite la evaluación de validez de elementos y atributos con esquemas. *Ir a la descripción*

La clase PSVI ofrece estas **constantes**:

PSVI.NOTKNOWN
PSVI.VALID
PSVI.INVALID
PSVI.NONE
PSVI.FULL
PSVI.PARTIAL
PSVI.SIMPLE
PSVI.COMPLEX

La clase PSVI ofrece estos **atributos de instancia** (solo lectura):

PSVI.ID_IDREF_table
Ver xsd.ID_IDREF_table.

La clase PSVI ofrece estos **métodos de instancia**:

PSVI.element()
Ofrece un elemento para la evaluación de validez del esquema. Ver también xsd.ElementPSVI.

PSVI.attribute()
Ofrece un atributo para la evaluación del validez del esquema. Ver también xsd.AttributePSVI.

 © 2013 Altova GmbH

5.7.35 xsd.Particle

clase **xsd.Particle**

La clase `Particle` contiene los componentes que contiene directa o indirectamente. Contiene directamente el componente que es el valor de su atributo `term`. Contiene indirectamente partículas, grupos, comodines y declaraciones de elemento que están en el valor de su propiedad `term`. *Ir a la descripción*

La clase `Particle` ofrece estos **atributos de instancia** (solo lectura):

Particle.min_occurs
Un valor `xs:nonNegativeInteger`. Necesario.

Particle.max_occurs
Un entero positivo o `unbounded`. Necesario.

Particle.term
Un componente `Term`. Necesario.

Particle.annotations
Una secuencia de componentes `Annotation`.

5.7.36 xsd.QName

clase **xsd.QName**
La clase QName representa un nombre XML completo. *Ir a la descripción*

La clase QName ofrece estos **atributos de instancia** (solo lectura):

QName.namespace_name
El nombre de la parte de espacio de nombres del nombre completo.

QName.local_part
La parte local del nombre completo.

 © 2013 Altova GmbH

5.7.37 xsd.Schema

clase **xsd.Schema**
La clase schema contiene una colección de componentes de esquema. P. ej. definiciones de tipo y declaraciones de elemento, que tienen un espacio de nombres de destino común. *Ir a la descripción*

La clase Schema ofrece estos **atributos de instancia** (solo lectura):

Schema.type_definitions
Un conjunto de componentes de definición de tipo. Podría ser una SimpleTypeDefinition o una ComplexTypeDefinition.

Schema.attribute_declarations
Un conjunto de componentes AttributeDeclaration.

Schema.element_declarations
Un conjunto de componentes ElementDeclaration.

Schema.attribute_group_definitions
Un conjunto de componentes AttributeGroupDefinition.

Schema.model_group_definitions
Un conjunto de componentes ModelGroupDefinition.

Schema.notation_declarations
Un conjunto de componentes NotationDeclaration.

Schema.identity_constraint_definitions
Un conjunto de componentes IdentityConstraintDefinition.

La clase Schema ofrece estos **métodos de instancia**:

Schema.resolve_type_definition()
Ofrece definiciones de tipo.

Schema.resolve_attribute_declaration()
Ofrece declaraciones de atributo.

Schema.resolve_element_declaration()
Ofrece declaraciones de elemento.

Schema.resolve_attribute_group_definition()
Ofrece definiciones de grupo de atributos.

Schema.resolve_model_group_definition()
Ofrece definiciones de modelo de grupo.

Schema.resolve_notation_declaration()
Ofrece declaraciones de notación.

Schema.resolve_identity_constraint_definition()
Ofrece definiciones de restricción de identidad.

5.7.38 xsd.Scope

clase **xsd.Scope**
La clase `Scope` representa un registro de la propiedad `Scope`. Necesario. *Ir a la descripción*

La clase `Scope` ofrece estas **constantes**:

Scope.GLOBAL
Scope.LOCAL

La clase `Scope` ofrece estos **atributos de instancia** (solo lectura):

Scope.variety
Uno de estos: `{global, local}`. Necesario.

Scope.parent
Bien un `ComplexTypeDefinition` o un `AttributeGroupDefinition`. Necesario si `variety` es `local`. De lo contrario, debe estar ausente.

 © 2013 Altova GmbH

5.7.39 xsd.Sibling

clase **xsd.Sibling**

La clase `Sibling` representa una palabra clave miembro de un conjunto de valores permitidos para el atributo `disallowed_names` de `NamespaceConstraint`. *Ir a la descripción*

La clase `Sibling` ofrece estos **métodos de instancia**:

Sibling.__str__()

5.7.40 xsd.SimpleTypeDefinition

clase **xsd.SimpleTypeDefinition**
La clase `SimpleTypeDefinition` representa tipos simples identificados por sus atributos `name` y `target namespace`. *Ir a la descripción*

La clase `SimpleTypeDefinition` ofrece estas **constantes**:

SimpleTypeDefinition.ATOMIC
SimpleTypeDefinition.LIST
SimpleTypeDefinition.UNION

La clase `SimpleTypeDefinition` ofrece estos **atributos de instancia** (solo lectura):

SimpleTypeDefinition.annotations
Una secuencia de componentes `Annotation`.

SimpleTypeDefinition.name
Un valor `xs:NCName`. Opcional.

SimpleTypeDefinition.target_namespace
Un valor `xs:anyURI`. Opcional.

SimpleTypeDefinition.context
Necesario si está ausente el atributo de instancia `name` (*ver más arriba*). De lo contrario, debe estar ausente. Puede ser `AttributeDeclaration`, `ElementDeclaration`, `ComplexTypeDefinition` o `SimpleTypeDefinition`.

SimpleTypeDefinition.base_type_definition
Un componente definición de tipo. Necesario.

SimpleTypeDefinition.facets
Un conjunto de componentes `ConstrainingFacet`.

SimpleTypeDefinition.final
Un subconjunto de `{extension, restriction, list, union}`.

SimpleTypeDefinition.variety
Uno de estos: `{atomic, list, union}`. Necesario para todas las definiciones de tipo simple excepto `xs:anySimpleType`, en la que está ausente.

SimpleTypeDefinition.primitive_type_definition
Un componente de definición de tipo simple. Con una excepción, necesario si `variety` es valor atómico. De lo contrario debe estar ausente. La excepción es `xs:anyAtomicType`, cuya definición `primitive_type_definition` está ausente. Si no está ausente, debe ser una definición primitiva.

SimpleTypeDefinition.item_type_definition
Un componente de definición de tipo simple. Necesario si `variety` es `list`. De lo contrario debe estar ausente. El valor de esta propiedad debe ser una definición de tipo simple normal o primitiva con `variety=atomic` o una definición de tipo simple normal con `variety=union` cuyos miembros básicos son todos atómicos. El valor no debe ser un tipo de lista (con `variety=list`) ni tener miembros básicos que son tipos de lista.

 © 2013 Altova GmbH

SimpleTypeDefinition.member_type_definitions
Una secuencia de componentes SimpleTypeDefinition primitivos o normales. Debe estar presente (pero puede estar vacío) si variety=union, . De lo contrario debe estar ausente. La secuencia puede contener cualquier definición de tipo simple primitiva o normal. Pero no puede contener ninguna definición de tipo especial.

5.7.41 xsd.TypeAlternative

clase **xsd.TypeAlternative**

La clase `TypeAlternative` es utilizada por una `ElementDeclaration` para especificar una condición (`test`) bajo la cual se utiliza un tipo determinado (`type_definition`) como definición de tipo de control para elementos de información `element` controlados por dicha `ElementDeclaration`. Cada `ElementDeclaration` puede tener varias `TypeAlternatives` en su `TypeTable`. *Ir a la descripción*

La clase `TypeAlternative` ofrece estos **atributos de instancia** (solo lectura):

TypeAlternative.annotations

Una secuencia de componentes `Annotation`.

TypeAlternative.test

Un registro de la propiedad `XPathExpression` que se utiliza para especificar una condición para seleccionar la declaración de tipo de control de una declaración de elemento. Opcional.

TypeAlternative.type_definition

Un componente de definición de tipo (`xsd.ComplexTypeDefnition` o `xsd.SimpleTypeDefinition`). Necesario.

 © 2013 Altova GmbH

5.7.42 xsd.TypeTable

clase **`xsd.TypeTable`**

La definición de tipo con la que se valida un elemento de información `element` (su definición de tipo de control) puede ser diferente a la definición de tipo declarada. La propiedad `TypeTable` de un `ElementDeclaration`, que controla la asignación de tipo condicional, y el atributo `xsi:type` de un elemento de información `element` pueden provocar que la definición de tipo de control y la definición de tipo declarada no sean iguales. *Ir a la descripción*

La clase `TypeTable` ofrece estos **atributos de instancia** (solo lectura):

`TypeTable.alternatives`
Una secuencia de componentes `TypeAlternative`.

`TypeTable.default_type_definition`
Un componente `TypeAlternative`. Necesario.

5.7.43 xsd.Unbounded

clase **xsd.Unbounded**

La clase `Unbounded` es un valor de cadena. Representa el valor superior de la propiedad `maxOccurs`. *Ir a la descripción*

La clase `Unbounded` ofrece estos **métodos de instancia**:

Unbounded.__str__()

 © 2013 Altova GmbH

5.7.44 xsd.ValueConstraint

clase **xsd.ValueConstraint**
La clase `ValueConstraint` representa una propiedad de la clase `AttributeUse`. *Ir a la descripción*

La clase `ValueConstraint` ofrece estas **constantes**:

ValueConstraint.DEFAULT
ValueConstraint.FIXED

La clase `ValueConstraint` ofrece estos **atributos de instancia** (solo lectura):

ValueConstraint.variety
Uno de estos: `{default, fixed}`. Necesario.

ValueConstraint.value
Un valor real. Necesario.

ValueConstraint.lexical_form
Una cadena de caracteres. Necesario.

5.7.45 xsd.XPathExpression

clase **xsd.XPathExpression**

Para revisar una aserción, se construye una instancia del modelo de datos XPath 2.0, en la que el elemento de información `element` que se evalúa es el nodo raíz (sin primarios) y a los elementos y atributos se les asignan tipos y valores de acuerdo con las reglas de construcción del modelo de datos XPath 2.0. Cuando se evalúa con esta instancia del modelo de datos, `test` da como resultado `true` o `false`. *Ir a la descripción*

La clase `XPathExpression` ofrece estos **atributos de instancia** (solo lectura):

XPathExpression.namespace_bindings
Un conjunto de registros de la propiedad `NamespaceBinding` para la expresión XPath.

XPathExpression.default_namespace
Un valor `xs:anyURI`. Opcional.

XPathExpression.base_URI
Un valor `xs:anyURI`. Opcional. El URI base para rutas de acceso relativas en la expresión XPath.

XPathExpression.expression
Una expresión XPath 2.0. Necesario.

 © 2013 Altova GmbH

5.7.46 Objetos de tipos de datos especiales integrados

Estos son los objetos de tipos de datos especiales integrados que están disponibles. Para más información sobre tipos de datos, consulte su descripción en los apartados *Special Built-in Datatypes* y *Primitive Datatypes* de la especificación XML Schema.

---

clase **xsd.anyAtomicType**
La clase `anyAtomicType` representa una restricción de `anySimpleType` y es el tipo base de los tipos primitivos.

clase **xsd.anySimpleType**
La clase `anySimpleType` representa una restricción de `anyType` y es el tipo base de `anyAtomicType`.

clase **xsd.anyURI**
La clase `anyURI` representa una referencia de Identificador de Recursos Internacionalizado (IRI). Su valor puede ser absoluto o relativo. Tiene un solo atributo de instancia de solo lectura: **anyURI.value**.

5.7.47 Objetos de tipo de datos string

La tabla siguiente muestra los objetos de tipo de datos `string` que están disponibles y sus atributos de instancia de solo lectura.

Clase	Atributos de instancia (solo lectura)
`xsd.language`	`language.value`
`xsd.normalizedString`	`normalizedString.value`
`xsd.string`	`string.value`
`xsd.token`	`token.value`

Para más información sobre tipos de datos, consulte la descripción de las secciones *Primitive Datatypes* y *Other Built-in Datatypes* de la especificación XML Schema.

© 2013 Altova GmbH

5.7.48 Objeto de tipo de datos boolean

clase **xsd.boolean**
Un objeto `boolean` representa un documento de instancia XBRL. Ofrece el atributo de instancia de solo lectura **boolean.value**, que devuelve un valor booleano. Para más información sobre tipos de datos, consulte su descripción en la sección *Primitive Datatypes* de la especificación XML Schema.

5.7.49 Objetos de tipo de datos number

Estos son los objetos de tipo de datos number que están disponibles. Todos tienen el atributo de instancia de solo lectura **`value`**, cuya representación léxica es diferente, dependiendo del objeto.

Clase	Atributos de instancia (solo lectura)
`xsd.byte`	`byte.value`
`xsd.decimal`	`decimal.value`
`xsd.double`	`double.value`
`xsd.float`	`float.value`
`xsd.int`	`int.value`
`xsd.integer`	`integer.value`
`xsd.long`	`long.value`
`xsd.negativeInteger`	`negativeInteger.value`
`xsd.nonNegativeInteger`	`nonNegativeInteger.value`
`xsd.nonPositiveInteger`	`nonPositiveInteger.value`
`xsd.positiveInteger`	`positiveInteger.value`
`xsd.short`	`short.value`
`xsd.unsignedByte`	`unsignedByte.value`
`xsd.unsignedInt`	`unsignedInt.value`
`xsd.unsignedLong`	`unsignedLong.value`
`xsd.unsignedShort`	`unsignedShort.value`

Para más información sobre este tipo de datos, consulte su descripción en la sección *Primitive Datatypes* y *Other Built-in Datatypes* de la especificación XML Schema.

© 2013 Altova GmbH

5.7.50 Objetos de tipo de datos duration

La tabla siguiente muestra los objetos de tipo de datos `duration` que están disponibles y sus atributos de instancia de solo lectura.

Clase	Atributos de instancia (solo lectura)
`xsd.dayTimeDuration`	`dayTimeDuration.months` `dayTimeDuration.seconds`
`xsd.duration`	`duration.months` `duration.seconds`
`xsd.yearMonthDuration`	`yearMonthDuration.months` `yearMonthDuration.seconds`

Para más información sobre este tipo de datos, consulte su descripción en la sección *Primitive Datatypes* y *Other Built-in Datatypes* de la especificación XML Schema.

5.7.51 Objetos de tipo de datos date y time

La tabla siguiente muestra los objetos de tipo de datos `date` y `time` que están disponibles y sus atributos de instancia de solo lectura. Si existe un atributo `value`, está compuesto de fragmentos disponibles como otros atributos del objeto. Por ejemplo: `time.value` está formado por los fragmentos `time.hour`, `time.minute`, `time.second` y `time.timezoneOffset`.

Clase	Atributos de instancia (solo lectura)
`xsd.date`	`dayTimeDuration.months` `dayTimeDuration.seconds`
`xsd.dateTime`	`duration.months` `duration.seconds`

`xsd.dateTimeStamp`	`dateTimeStamp.value`
	`dateTimeStamp.year`
	`dateTimeStamp.month`
	`dateTimeStamp.day`
	`dateTimeStamp.hour`
	`dateTimeStamp.minute`
	`dateTimeStamp.second`
	`dateTimeStamp.timezoneOffset`

`xsd.gDay`	`gDay.day` `gDay.timezoneOffset`
`xsd.gMonth`	`gMonth.month` `gMonth.timezoneOffset`

`xsd.gMonthDay`	`gMonthDay.month`
	`gMonthDay.days`
	`gMonthDay.timezoneOffset`

`xsd.gYear`	`gYear.year` `gYear.timezoneOffset`

`xsd.gYearMonth`	`gYearMonth.year`
	`gYearMonth.month`
	`gYearMonth.timezoneOffset`

`xsd.time`	`time.value`
	`time.hour`
	`time.minute`
	`time.second`
	`time.timezoneOffset`

Para más información sobre este tipo de datos, consulte su descripción en la sección *Primitive Datatypes* y *Other Built-in Datatypes* de la especificación XML Schema.

© 2013 Altova GmbH

5.7.52 Objetos de tipo de datos binary

La tabla siguiente muestra los objetos de tipo de datos `binary` que están disponibles y sus atributos de instancia de solo lectura.

Clase	Atributos de instancia (solo lectura)
`xsd.base64Binary`	`base64Binary.value`
`xsd.hexBinary`	`hexBinary.value`

Para más información sobre este tipo de datos, consulte su descripción en la sección *Primitive Datatypes* de la especificación XML Schema.

5.7.53 Objetos faceta

Los tipos de datos derivados por restricción también pueden tener las facetas de restricción que permita la especificación. A continuación puede ver los objetos faceta disponibles. La tabla enumera los objetos faceta que tienen atributos de instancia de solo lectura. Los objetos `xsd.explicitTimezoneFacet` y `xsd.whiteSpaceFacet` tienen constantes además de atributos de instancia de solo lectura (*ver lista después de la tabla*).

Para más información sobre las facetas, consulte su descripción en la sección Constraining Facets de la especificación XML Schema (en la tabla haga clic en el enlace para ver la descripción de la faceta en la especificación).

Clase	Atributos de instancia (solo lectura)
xsd.assertionsFacet	assertionsFacet.annotations assertionsFacet.value
xsd.enumerationFacet	enumerationFacet.annotations enumerationFacet.value
xsd.fractionDigitsFacet	fractionDigitsFacet.annotations fractionDigitsFacet.value
xsd.lengthFacet	lengthFacet.annotations lengthFacet.value lengthFacet.fixed
xsd.maxExclusiveFacet	maxExclusiveFacet.annotations maxExclusiveFacet.value maxExclusiveFacet.fixed
xsd.maxInclusiveFacet	maxInclusiveFacet.annotations maxInclusiveFacet.value maxInclusiveFacet.fixed
xsd.maxLengthFacet	maxLengthFacet.annotations maxLengthFacet.value maxLengthFacet.fixed
xsd.minExclusiveFacet	minExclusiveFacet.annotations minExclusiveFacet.value minExclusiveFacet.fixed
xsd.minInclusiveFacet	minInclusiveFacet.annotations minInclusiveFacet.value minInclusiveFacet.fixed
xsd.minLengthFacet	minLengthFacet.annotations minLengthFacet.value minLengthFacet.fixed
xsd.pattern	patternFacet.annotations patternFacet.value
xsd.totalDigitsFacet	totalDigitsFacet.annotations totalDigitsFacet.value totalDigitsFacet.fixed

`xsd.explicitTimezoneFacet`

- *Constantes*:
  ```
  explicitTimezoneFacet.REQUIRED
  explicitTimezoneFacet.PROHIBITED
  explicitTimezoneFacet.OPTIONAL
  ```
- *Atributos de instancia de solo lectura*:

© 2013 Altova GmbH

```
whiteSpaceFacet.annotations
whiteSpaceFacet.value
whiteSpaceFacet.fixed
```

xsd.whiteSpaceFacet

- *Constantes*:
  ```
  whiteSpaceFacet.PRESERVE
  whiteSpaceFacet.REPLACE
  whiteSpaceFacet.COLLAPSE
  ```
- *Atributos de instancia de solo lectura*:
  ```
  whiteSpaceFacet.annotations
  whiteSpaceFacet.value
  whiteSpaceFacet.fixed
  ```

Altova RaptorXML 2013

Interfaz Java

6 Interfaz Java

La API de Java viene empaquetada en el paquete `com.altova.raptorxml`. El punto de entrada es la interfaz RaptorXMLFactory, que ofrece métodos para obtener objetos del motor para validación y transformación.

Los métodos getter siempre devuelven objetos nuevos. Para crear una instancia de RaptorXMLFactory debe llamar a RaptorXML.getFactory().

[Este método de generador devuelve un generador para el producto RaptorXMLServer y otro generador distinto para el producto RaptorXMLDev).

La interfaz pública de RaptorXMLFactory se describe con este código:

```
public interface RaptorXMLFactory
{
   public XMLValidator getXMLValidator();
   public XSLT getXBRL();
   public XQuery getXQuery();
   public XSLT getXSLT();
   public void setServerName( String nombre ) throws RaptorXMLException;
   public void setServerFile( String file ) throws RaptorXMLException;
   public void setServerPort( int port ) throws RaptorXMLException;
     public void setUserCatalog(String catalog);
     public void setGlobalResourcesFile(String file);
     public void setGlobalResourceConfig(String config);
     public void setErrorLimit(int limit);
     public void setReportOptionalWarnings(boolean report);
```

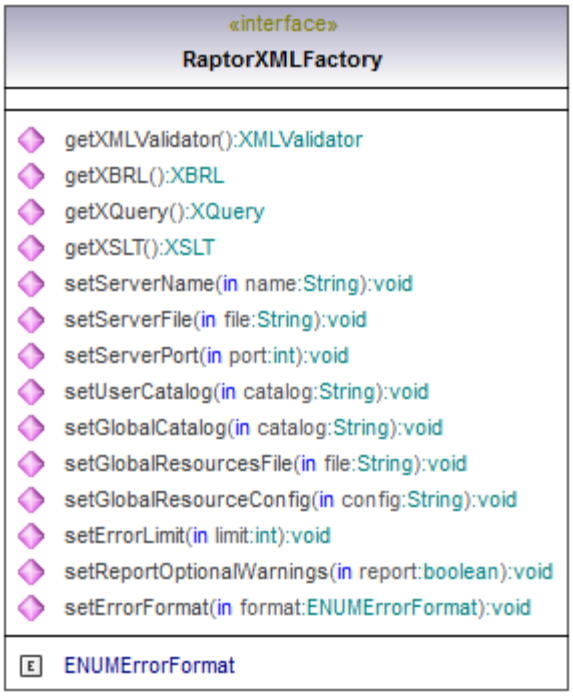
Las clases de `com.altova.engines` aparecen enumeradas a continuación y se describen en los apartados siguientes.

- RaptorXMLFactory
  Crea una instancia nueva del objeto servidor COM de RaptorXML mediante llamada nativa y ofrece acceso a los motores de RaptorXML.
- XMLValidator
  Clase que contiene el validador XML.
- XBRL
  Clase que contiene el validador XBRL.
- XSLT
  Clase que contiene los motores XSLT 1.0, 2.0, 3.0.
- XQuery
  Clase que contiene los motores XQuery 1.0, 3.0.

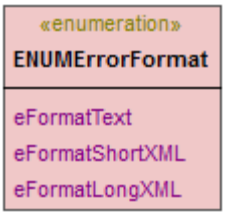
© 2013 Altova GmbH

6.1 RaptorXMLJava - RaptorXMLFactory

Interfaz **RaptorXMLFactory**

diagrama	
documentación	Use RaptorXMLFactory() para crear una instancia nueva del objeto servidor COM RaptorXML. Esto ofrece acceso al motor RaptorXML. RaptorXMLFactory y el objeto COM RaptorXML COM tienen una relación uno a uno.

Enumeración **RaptorXMLFactory::ENUMErrorFormat**

diagrama	
elementosConTipo	Interfaz **RaptorXMLFactory** Operación **setErrorFormat**
documentación	Contiene los literales de enumeración que definen el formato de errores de salida.

LiteralDeEnumeración **RaptorXMLFactory::ENUMErrorFormat::eFormatLongXML**

documentación	Establece el formato de salida XML largo para los errores. Este formato de salida XML es el que ofrece más detalles.

LiteralDeEnumeración **RaptorXMLFactory::ENUMErrorFormat::eFormatShortXML**

documentación	Establece el formato de salida XML breve para los errores. Este formato de salida XML es más breve que el formato XML largo.

LiteralDeEnumeración **RaptorXMLFactory::ENUMErrorFormat::eFormatText**

documentación	Establece el formato de salida texto para los errores. Es la configuración predeterminada.

Operación **RaptorXMLFactory::getXBRL**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	return	**return**	**XBRL**		
documentación	Recupera el motor XBRL y devuelve una instancia XBRL nueva de RaptorXMLFactory. Solo compatible con RaptorXML+XBRL Server. Parámetros: ninguno				

Operación **RaptorXMLFactory::getXMLValidator**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	return	**return**	**XMLValidator**		
documentación	Recupera el motor XML y devuelve una instancia nueva del validador XML de RaptorXMLFactory. Parámetros: ninguno				

Operación **RaptorXMLFactory::getXQuery**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	return	**return**	**XQuery**		
documentación	Recupera el motor XQuery y devuelve una instancia nueva del validador XQuery de RaptorXMLFactory. Parámetros: ninguno				

Operación **RaptorXMLFactory::getXSLT**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	return	**return**	**XSLT**		
documentación	Recupera el motor XSLT y devuelve una instancia nueva del validador XSLT de RaptorXMLFactory. Parámetros: ninguno				

© 2013 Altova GmbH

Operación **RaptorXMLFactory::setErrorFormat**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	format	**in**	**ENUMErrorFormat**		
	return	**return**	**void**		
documentación	Establece el formato de los errores de RaptorXML (texto, XML breve, XML largo), dependiendo de los literales ENUMErrorFormat. Parámetros: 'format' contiene el valor del literal de enumeración seleccionado, de tipo ENUMErrorFormat.				

Operación **RaptorXMLFactory::setErrorLimit**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	limit	**in**	**int**		
	return	**return**	**void**		
documentación	Configura la propiedad de límite de errores de validación de RaptorXML. Propiedades: 'limit' define el número de errores que se deben comunicar antes de detener la ejecución, de tipo int. Use -1 para definir un número ilimitado de errores, es decir, se comunicarán todos los errores de validación.				

Operación **RaptorXMLFactory::setGlobalCatalog**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	catalog	**in**	**String**		
	return	**return**	**void**		
documentación	Establece el nombre del archivo (como URL) del catálogo global (p. ej. RootCatalog.xml). Parámetros: "catalog" es la URL de la ubicación base del archivo de catálogo global, de tipo string. Nota: use la URL absoluta para el nombre de archivo de entrada. Una URL absoluta especifica la ubicación exacta del archivo p. ej. http://www.miSitioWeb.com/archivosxml/miCatGlobal.xml				

Operación **RaptorXMLFactory::setGlobalResourceConfig**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	config	**in**	**String**		
	return	**return**	**void**		
documentación	Establece la configuración activa del recurso global. Parámetros: 'config' es el nombre de la configuración usada por el recurso global activo, de tipo string.				

Operación **RaptorXMLFactory::setGlobalResourcesFile**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	file	**in**	**String**		
	return	**return**	**void**		
documentación	Establece el nombre de archivo / la URL del archivo XML de recursos globales (p. ej. RecursosGlobales.xml). Parámetros: 'file' contiene el nombre del archivo XML, de tipo string. Nota: use la URL absoluta para el nombre del archivo de entrada. Una URL absoluta especifica la ubicación exacta del archivo p. ej. http://www.miSitioWeb.com/archivosxml/miEntradaxml.xml				

Operación **RaptorXMLFactory::setReportOptionalWarnings**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	report	**in**	**boolean**		
	return	**return**	**void**		
documentación	Habilita / deshabilita la emisión de mensajes de advertencia. Parámetros: 'report' contiene el valor booleano, de tipo boolean. Valores: 'true' habilita la operación, 'false' la deshabilita.				

Operación **RaptorXMLFactory::setServerFile**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	file	**in**	**String**		
	return	**return**	**void**		
documentación	Establece la dirección del servidor para el servidor HTTP (este método falla en la edición RaptorXML Development Edition). Parámetros: 'file' es la dirección del servidor (relativa a la raíz del servidor HTTP), de tipo string. Genera una excepción RaptorXMLException cuando se produce un error.				

Operación **RaptorXMLFactory::setServerName**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	nombre	**in**	**String**		
	return	**return**	**void**		
documentación	Establece el nombre del servidor para el servidor HTTP (este método falla en la edición RaptorXML Development Edition). Parámetros: 'name' es el nombre del servidor, de tipo string. Genera una excepción RaptorXMLException cuando se produce un error.				

 © 2013 Altova GmbH

Operación **RaptorXMLFactory::setServerPort**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	port	**in**	**int**		
	return	**return**	**void**		
documentación	Establece el puerto del servidor para el servidor HTTP (este método falla en la edición RaptorXML Development Edition). Parámetros: 'port' es el puerto del servidor, de tipo int. Genera una excepción RaptorXMLException cuando se produce un error.				

Operación **RaptorXMLFactory::setUserCatalog**

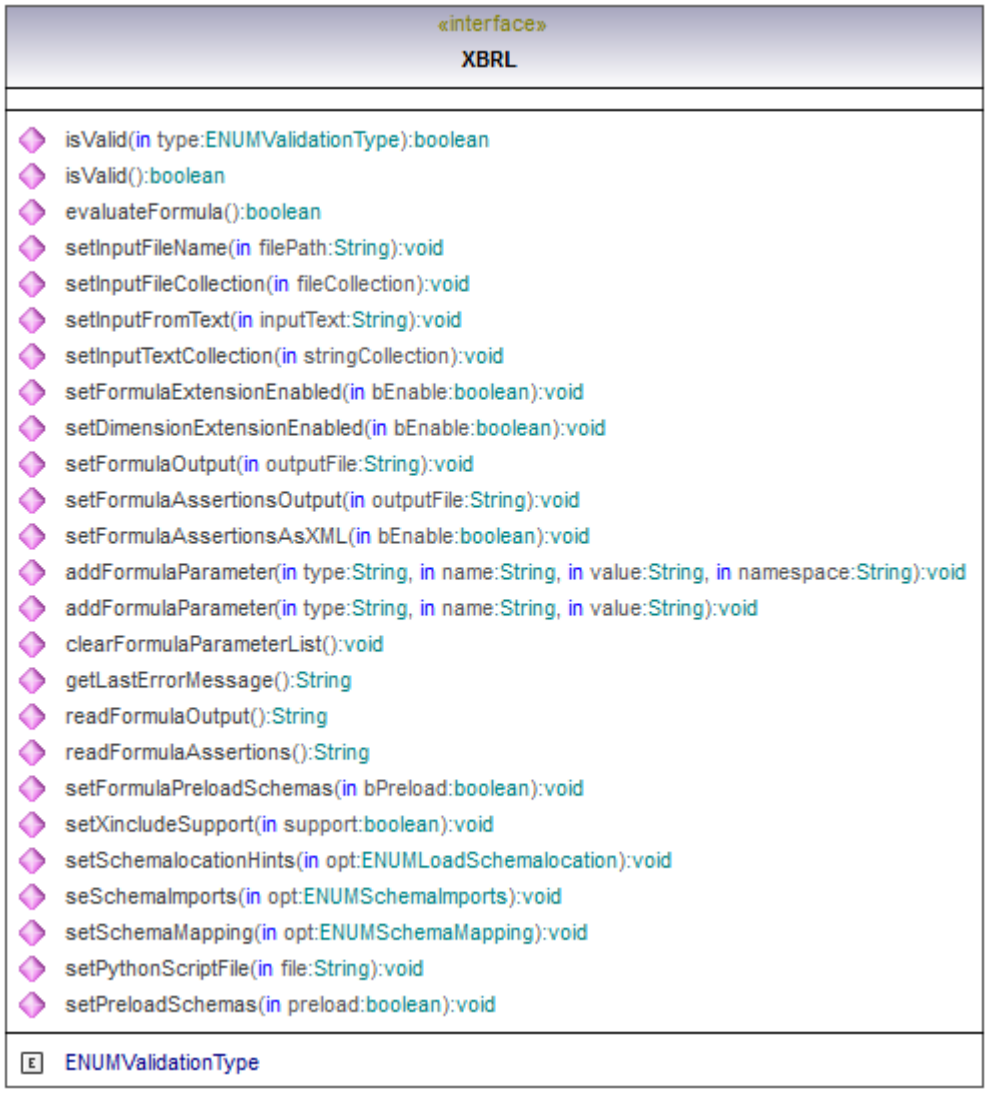
parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	catalog	**in**	**String**		
	return	**return**	**void**		
documentación	Establece el nombre (como URL) para el catálogo definido por el usuario (p. ej. CustomCatalog.xml). Parámetros: 'catalog' es el nombre del catálogo definido por el usuario, de tipo string. Nota: use la URL absoluta para el nombre de archivo de entrada. Una URL absoluta especifica la ubicación exacta del archivo..				

Documentación UML generada con el editor UML **UModel**. **Más información en http://www.altova.com/es/umodel** 06/07/13 15:46:38

6.2 RaptorXMLJava - XBRL

Interfaz **XBRL**

diagrama



elementosConTipo

Interfaz **RaptorXMLFactory** Operación **getXBRL**

documentación

Valida el documento XML/XBRL dado (Nota: XBRL solo es compatible con RaptorXML+XBRL) con DTD o esquemas XML internos o externos.

Comprueba si los documentos XML, DTD, XML Schema, XSLT y XQuery tienen un formato correcto.

Valida taxonomías XBRL y documentos XBRL con taxonomías XBRL (solo en RaptorXML+XBRL).

Operación **XBRL::addFormulaParameter**

 © 2013 Altova GmbH

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	tipo	**in**	**String**		
	nombre	**in**	**String**		
	value	**in**	**String**		
	namespace	**in**	**String**		
	return	**return**	**void**		
documentación	Añade un parámetro utilizado en el proceso de evaluación de fórmulas. Parámetros: 'type' contiene el tipo de datos del parámetro, de tipo string. 'name' contiene el nombre del parámetro, de tipo string. 'value' contiene el valor del parámetro, de tipo string. 'namespace' contiene el espacio de nombres del parámetro, de tipo string.				

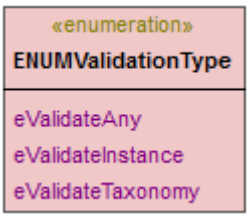
Operación **XBRL::addFormulaParameter**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	tipo	**in**	**String**		
	nombre	**in**	**String**		
	value	**in**	**String**		
	return	**return**	**void**		
documentación	Añade un parámetro utilizado en el proceso de evaluación de fórmulas. Parámetros: 'type' contiene el tipo de datos del parámetro, de tipo string. 'name' contiene el nombre del parámetro, de tipo string. 'value' contiene el valor del parámetro, de tipo string.				

Operación **XBRL::clearFormulaParameterList**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	return	**return**	**void**		
documentación	Borra la lista de parámetros de la fórmula creada con el método addFormulaParameter. Parámetros: ninguno.				

Enumeración **XBRL::ENUMValidationType**

diagrama	
elementosConTipo	Interfaz **XBRL** Operación **isValid**
documentación	Contiene literales de enumeración definiendo el tipo de documento que se validará.

LiteralDeEnumeración **XBRL::ENUMValidationType::eValidateAny**

documentación	Valida documentos XBRL (ya sean de instancia o taxonomía). Este comando también se usa para validar documentos XML, DTD, XSD, XSLT o XQuery. El tipo de documento suministrado se detecta automáticamente.

LiteralDeEnumeración **XBRL::ENUMValidationType::eValidateInstance**

documentación	Valida un documento de instancia XBRL (extensión `.xbrl`).

LiteralDeEnumeración **XBRL::ENUMValidationType::eValidateTaxonomy**

documentación	Valida un documento de taxonomía (esquema) XBRL (extensión `.xsd`).

Operación **XBRL::evaluateFormula**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	return	**return**	**boolean**		
documentación	Evalúa fórmulas XBRL en un archivo de instancia XBRL. Devuelve: 'true' cuando la operación finaliza correctamente y 'false' cuando hay un fallo. Si se produce un error durante la ejecución, use la operación getLastErrorMessage para obtener información adicional. Genera una excepción RaptorXMLException cuando se produce un error.				

Operación **XBRL::getLastErrorMessage**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	return	**return**	**String**		
documentación	Recupera el último mensaje de error del motor XBRL. Parámetros: ninguno Devuelve el texto del último mensaje de error, de tipo string.				

Operación **XBRL::isValid**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	tipo	**in**	**ENUMValidationType**		
	return	**return**	**boolean**		
documentación	Resultado de una validación XBRL con la taxonomía especificada con el valor de ENUMValidationType. Parámetros: 'type' contiene el valor del literal de enumeración. Genera una excepción RaptorXMLException cuando se produce un error. Si se produce un error durante la ejecución, use la operación getLastErrorMessage para obtener información adicional.				

© 2013 Altova GmbH

Operación **XBRL::isValid**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	return	**return**	**boolean**		
documentación	Resultado de la validación XBRL con la taxonomía. Parámetros: ninguno. Valores: 'true' si la operación finaliza correctamente, 'false' si falla. Genera una excepción RaptorXMLException cuando se produce un error. Si se produce un error durante la ejecución, use la operación getLastErrorMessage para obtener información adicional.				

Operación **XBRL::readFormulaAssertions**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	return	**return**	**String**		
documentación	Lee la evaluación de la evaluación de aserción de fórmula del archivo especificado. Parámetros: ninguno.				

Operación **XBRL::readFormulaOutput**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	return	**return**	**String**		
documentación	Lee el resultado de la evaluación de aserción de fórmula del archivo especificado. Parámetros: ninguno.				

Operación **XBRL::setSchemaImports**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	opt	**in**	**ENUMSchemaImports**		
	return	**return**	**void**		
documentación	Establece el formato de los errores de RaptorXML (texto, XML breve, XML largo), dependiendo de los literales ENUMErrorFormat. Parámetros: 'format' contiene el valor del literal de enumeración seleccionado, de tipo ENUMErrorFormat.				

Operación **XBRL::setDimensionExtensionEnabled**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	bEnable	**in**	**boolean**		
	return	**return**	**void**		

documentación	Habilita la validación con XBRL Dimension. Parámetros: 'bEnable' contiene el valor booleano, de tipo boolean. Valores: 'true' habilita la validación con extensión para dimensiones. 'false' la deshabilita.

Operación **XBRL::setFormulaAssertionsAsXML**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	bEnable	**in**	**boolean**		
	return	**return**	**void**		
documentación	Permite dar formato XML al archivo de aserción cuando RaptorXML se ejecuta con las aserciones habilitadas. Parámetros: 'bEnable' contiene el valor booleano, de tipo boolean. Valores: 'true' habilita el formato de salida XML, 'false' genera el formato de salida JSON.				

Operación **XBRL::setFormulaAssertionsOutput**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	outputFile	**in**	**String**		
	return	**return**	**void**		
documentación	Establece la ubicación del archivo de salida de la aserción de la fórmula. Parámetros: 'outputFile' contiene la ruta completa del archivo de salida, de tipo string.				

Operación **XBRL::setFormulaExtensionEnabled**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	bEnable	**in**	**boolean**		
	return	**return**	**void**		
documentación	Habilita la validación con XBRL Formula. Parámetros: 'bEnable' contiene el valor booleano, de tipo boolean. Valores: 'true' habilita la validación con extensión para fórmulas. 'false' la deshabilita.				

Operación **XBRL::setFormulaOutput**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	outputFile	**in**	**String**		
	return	**return**	**void**		

© 2013 Altova GmbH

documentación	Establece la ubicación del archivo de evaluación de fórmulas XBRL. Parámetros: 'outputFile' es la ruta completa del archivo de salida.

Operación **XBRL::setFormulaPreloadSchemas**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	bPreload	**in**	**boolean**		
	return	**return**	**void**		
documentación	Define si los esquemas de XBRL Formula se cargarán previamente. Parámetros: 'bPreload' contiene el valor booleano, de tipo boolean. Valores: 'true' precarga los esquemas de la especificación XBRL Formula 1.0. El valor predeterminado es 'false'.				

Operación **XBRL::setInputFileCollection**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	fileCollection	**in**			
	return	**return**	**void**		
documentación	Establece la colección de archivos XBRL que se usará como instancias/datos de entrada. Parámetros: 'fileCollection' es una colección de cadenas que contiene las URL absolutas de cada uno de los archivos de entrada.				

Operación **XBRL::setInputFileName**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	filePath	**in**	**String**		
	return	**return**	**void**		
documentación	Establece le nombre de archivo para el archivo de instancia de entrada XBRL. Parámetros: 'filePath' es una cadena con la URL absoluta (o ubicación base) del archivo de entrada.				

Operación **XBRL::setInputFromText**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	inputText	**in**	**String**		
	return	**return**	**void**		
documentación	Suministra el contenido del documento de entrada XBRL como texto. Parámetros: 'inputText' contiene los datos XBRL, de tipo string.				

Operación **XBRL::setInputTextCollection**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	stringCollection	**in**			
	return	**return**	**void**		
documentación	Establece la colección de archivos que se usarán como datos de entrada. Parámetros: 'stringCollection' es una colección de cadenas que contiene las URL absolutas de cada uno de los archivos de instancia XBRL de entrada, de tipo string.				

Operación **XBRL::setPreloadSchemas**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	preload	**in**	**boolean**		
	return	**return**	**void**		
documentación	Define si los esquemas XBRL 2.1 se cargarán previamente. Parámetros: 'preload' contiene el valor booleano, de tipo boolean. Valores: 'true' precarga los esquemas de XBRL. El valor predeterminado es 'false'.				

Operación **XBRL::setPythonScriptFile**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	file	**in**	**String**		
	return	**return**	**void**		
documentación	Establece el nombre de archivo (como URL) del archivo del script Python. Parámetros: 'file' es una URL absoluta que da la ubicación base del archivo, de tipo string.				

Operación **XBRL::setSchemaLocationHints**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	opt	**in**	**ENUMLoadSchemalocation**		
	return	**return**	**void**		
documentación	Establece la ubicación desde la que se cargará el esquema, depende de ENUMLoadSchemaLocation. Parámetros: 'opt' contiene el valor del literal de enumeración seleccionado, de tipo ENUMLoadSchemaLocation.				

Operación **XBRL::setSchemaMapping**

© 2013 Altova GmbH

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	opt	**in**	**ENUMSchemaMapping**		
	return	**return**	**void**		
documentación	Establece la ubicación desde la que se cargará el esquema, depende de ENUMSchemaMapping. Parámetros: 'opt' contiene el valor del literal de enumeración seleccionado, de tipo ENUMSchemaMapping.				

Operación **XBRL::setXincludeSupport**

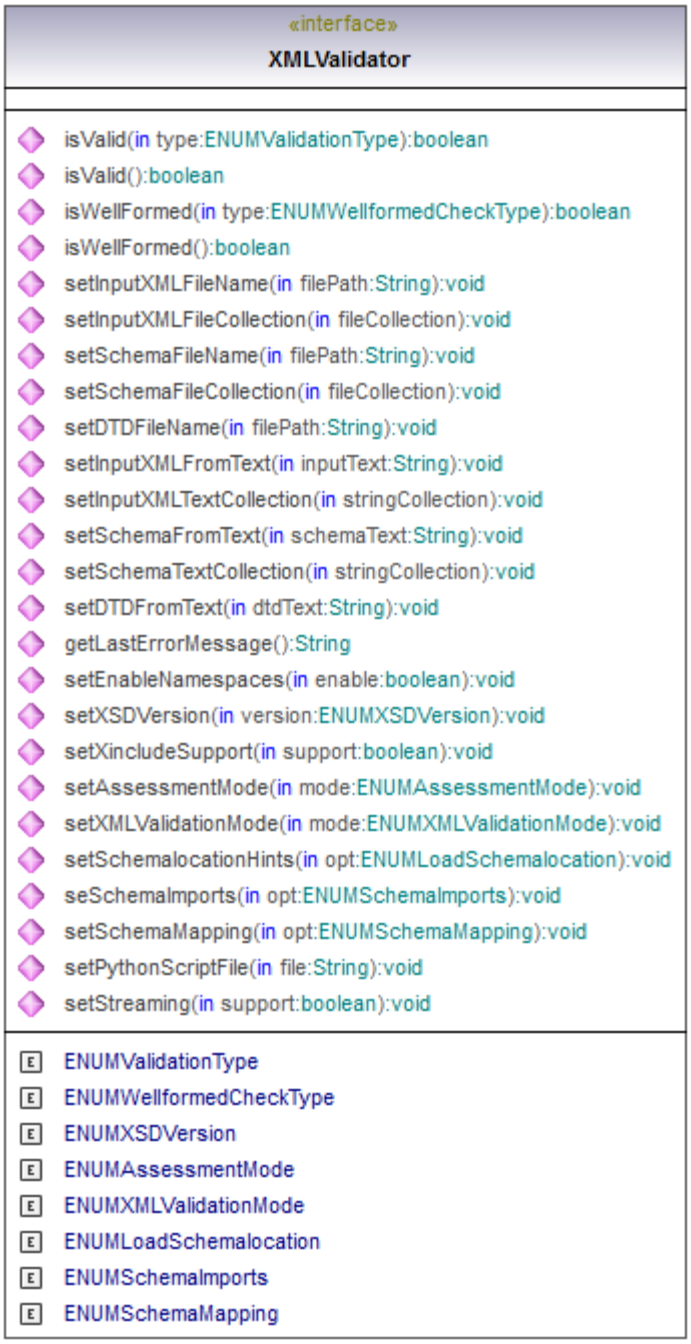
parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	support	**in**	**boolean**		
	return	**return**	**void**		
documentación	Habilita o deshabilita el uso de inclusiones XML (elementos XInclude). Parámetros: 'support' contiene el valor booleano, de tipo boolean. Valores: 'true' habilita los elementos XInclude include, 'false' los ignora.				

Documentación UML generada con el editor UML **UModel**. **Más información en http://www.altova.com/es/umodel** 06/07/13 15:46:38

6.3 RaptorXMLJava - XMLValidator

Interfaz **XMLValidator**

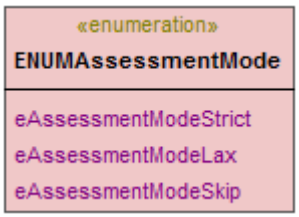
diagrama



elementosConTipo Interfaz **RaptorXMLFactory** Operación **getXMLValidator**

documentación	Valida el documento XML suministrado con esquemas XML o DTD internos o externos. Comprueba si los documentos XML, DTD, XML Schema, XSLT y XQuery tienen un formato correcto.

Enumeración **XMLValidator::ENUMAssessmentMode**

diagrama	
elementosConTipo	Interfaz **XMLValidator** Operación **setAssessmentMode**
documentación	Contiene literales de enumeración que definen el modo de evaluación del validador XML: Strict/Lax.

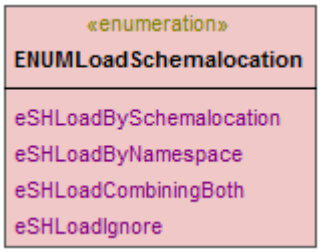
LiteralDeEnumeración **XMLValidator::ENUMAssessmentMode::eAssessmentModeLax**

documentación	Establece el modo de evaluación de validez del esquema en lax.

LiteralDeEnumeración **XMLValidator::ENUMAssessmentMode::eAssessmentModeStrict**

documentación	Establece el modo de evaluación de validez del esquema en strict.

Enumeración **XMLValidator::ENUMLoadSchemalocation**

diagrama	
elementosConTipo	Interfaz **XBRL** Operación **setSchemalocationHints** Interfaz **XMLValidator** Operación **setSchemalocationHints** Interfaz **XSLT** Operación **setSchemalocationHints**
documentación	Contiene los literales de enumeración que definen el comportamiento de loadSchemaLocation y cada literal tiene un atributo de espacio de nombres opcional y un atributo opcional schemaLocation.

LiteralDeEnumeración **XMLValidator::ENUMLoadSchemalocation::eSHLoadByNamespace**

documentación	Asigna LoadByNamespace a LoadSchemaLocation Utiliza la parte de espacio de nombres de xsi:schemaLocation y una cadena vacía en el caso de xsi:noNamespaceSchemaLocation y ubica el esquema mediante una asignación.

LiteralDeEnumeración **XMLValidator::ENUMLoadSchemalocation::eSHLoadBySchemalocation**

documentación	Asigna LoadBySchemalocation a LoadSchemaLocation. Usa la URL de la ubicación del esquema en los atributos xsi:schemaLocation y xsi:noNamespaceSchemaLocation de los documentos de instancia XML o XBRL. Este es el valor predeterminado.

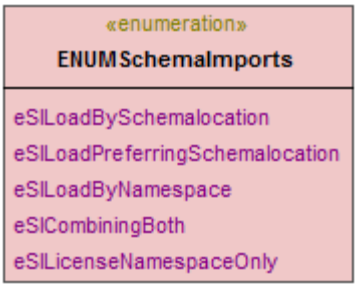
LiteralDeEnumeración **XMLValidator::ENUMLoadSchemalocation::eSHLoadCombiningBoth**

documentación	Asigna CombiningBoth a LoadSchemaLocation. Si el espacio de nombres o la URL tiene una asignación de catálogo, se utiliza la asignación. Si ambos tienen una asignación de catálogo, el valor del parámetro schema-mapping decide qué asignación se utiliza. Si ni el espacio de nombres ni la URL tienen una asignación de catálogo, se usa la URL.

LiteralDeEnumeración **XMLValidator::ENUMLoadSchemalocation::eSHLoadIgnore**

documentación	Asigna LoadIgnore a LoadSchemaLocation. Si el valor del parámetro es 'ignore', se ignoran los archivos xsi:schemaLocation y xsi:noNamespaceSchemaLocation.

Enumeración **XMLValidator::ENUMSchemaImports**

diagrama	
elementosConTipo	Interfaz **XBRL** — Operación **setSchemaImports** Interfaz **XMLValidator** — Operación **setSchemaImports** Interfaz **XSLT** — Operación **setSchemaImports**
documentación	Contiene los literales de enumeración que definen el comportamiento de los elementos xs:import y cada uno tiene un atributo de espacio de nombres opcional y un atributo schemaLocation opcional.

 © 2013 Altova GmbH

LiteralDeEnumeración **XMLValidator::ENUMSchemaImports::eSICombiningBoth**

documentación	Asigna CombiningBoth a SchemaImport. Si el espacio de nombres o la URL tiene una asignación de catálogo, se utiliza la asignación. Si ambos tienen una asignación de catálogo, el valor del parámetro schema-mapping decide qué asignación se utiliza. Si ni el espacio de nombres ni la URL tienen una asignación de catálogo, se usa la URL.

LiteralDeEnumeración **XMLValidator::ENUMSchemaImports::eSILicenseNamespaceOnly**

documentación	Asigna LicenseNamespaceOnly a SchemaImport. Se importa el espacio de nombres. No se importa el documento de esquema.

LiteralDeEnumeración **XMLValidator::ENUMSchemaImports::eSILoadByNamespace**

documentación	Asigna LicenseNamespaceOnly a SchemaImport. Se usa el valor del atributo de espacio de nombres para ubicar el esquema mediante una asignación de catálogo.

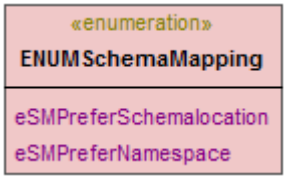
LiteralDeEnumeración **XMLValidator::ENUMSchemaImports::eSILoadBySchemalocation**

documentación	Asigna LoadBySchemalocation a SchemaImport. El valor del atributo de schemaLocation se utiliza para ubicar el esquema. Si está el atributo de espacio de nombres, se importa el espacio de nombres (con licencia).

LiteralDeEnumeración **XMLValidator::ENUMSchemaImports::eSILoadPreferringSchemalocation**

documentación	Asigna LoadPreferringSchemalocation a SchemaImport. Si está presente el atributo schemaLocation, el atributo se utiliza, teniendo en cuenta las asignaciones de catálogo. Si no está presente el atributo schemaLocation, entonces se usa el valor del atributo de espacio de nombres mediante una asignación de catálogo. Este es el valor predeterminado.

Enumeración **XMLValidator::ENUMSchemaMapping**

diagrama	
elementosConTipo	Interfaz **XBRL** — Operación **setSchemaMapping** Interfaz **XMLValidator** — Operación **setSchemaMapping** Interfaz **XSLT** — Operación **setSchemaMapping**

documentación	Contiene los literales de enumeración que definen cuál de las dos asignaciones de catálogo se utilizará.

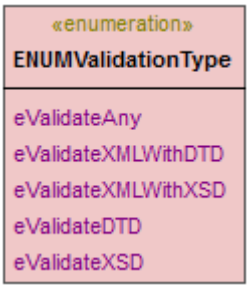
LiteralDeEnumeración **XMLValidator::ENUMSchemaMapping::eSMPreferNamespace**

documentación	Asigna PreferNamespace a SchemaMappingLocation.

LiteralDeEnumeración **XMLValidator::ENUMSchemaMapping::eSMPreferSchemaLocation**

documentación	Asigna PreferSchemaLocation a SchemaMappingLocation. El valor de PreferSchemaLocation remite a la asignación de URL.

Enumeración **XMLValidator::ENUMValidationType**

diagrama	
elementosConTipo	Interfaz **XMLValidator** Operación **isValid**
documentación	Contiene los literales de enumeración que definen cómo se validará el documento.

LiteralDeEnumeración **XMLValidator::ENUMValidationType::eValidateAny**

documentación	Establece el tipo de validación 'cualquiera'. Esto valida cualquier documento. El tipo de documento se detecta automáticamente.

LiteralDeEnumeración **XMLValidator::ENUMValidationType::eValidateDTD**

documentación	Establece el tipo de validación 'validar DTD'. Esto valida un documento DTD.

LiteralDeEnumeración **XMLValidator::ENUMValidationType::eValidateXMLWithDTD**

documentación	Establece el tipo de validación 'XML con DTD'. Esto valida un documento XML con un DTD.

LiteralDeEnumeración **XMLValidator::ENUMValidationType::eValidateXMLWithXSD**

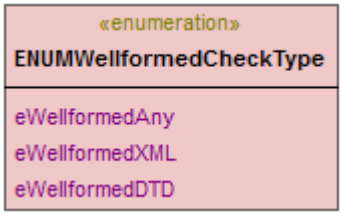
 © 2013 Altova GmbH

documentación	Establece el tipo de validación 'XML con XSD'. Esto valida un documento XML con un esquema XML.

LiteralDeEnumeración **XMLValidator::ENUMValidationType::eValidateXSD**

documentación	Establece el tipo de validación 'validar XSD'. Esto valida un documento de esquema XML del W3C.

Enumeración **XMLValidator::ENUMWellformedCheckType**

diagrama	
elementosConTipo	Interfaz **XMLValidator** Operación **isWellFormed**
documentación	Contiene los literales de enumeración que definen el tipo de comprobaciones de formato que se realizarán.

LiteralDeEnumeración **XMLValidator::ENUMWellformedCheckType::eWellformedAny**

documentación	Establece el tipo de comprobación de formato 'cualquiera' Comprueba si un documento XML o DTD tiene un formato correcto, según la especificación correspondiente. El tipo de documento se detecta automáticamente.

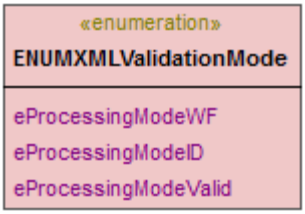
LiteralDeEnumeración **XMLValidator::ENUMWellformedCheckType::eWellformedDTD**

documentación	Establece el tipo de comprobación de formato 'DTD' Comprueba si los documentos DTD tienen un formato correcto según la especificación XML 1.0 o XML 1.1. Con RaptorXML Server y RaptorXML+XBRL Server puede realizar comprobaciones de formato en varios documentos a la vez.

LiteralDeEnumeración **XMLValidator::ENUMWellformedCheckType::eWellformedXML**

documentación	Establece el tipo de comprobación de formato 'XML' Comprueba si los documentos XML tienen un formato correcto según la especificación XML 1.0 o XML 1.1. Con RaptorXML Server y RaptorXML+XBRL Server puede realizar comprobaciones de formato en varios documentos a la vez.

Enumeración **XMLValidator::ENUMXMLValidationMode**

diagrama	
elementosConTipo	Interfaz **XMLValidator** Interfaz **XQuery** Interfaz **XSLT** Operación **setXMLValidationMode** Operación **setXMLValidationMode** Operación **setXMLValidationMode**
documentación	Contiene los literales de enumeración que definen el modo de procesamiento XML que se utilizará.

LiteralDeEnumeración **XMLValidator::ENUMXMLValidationMode::eProcessingModeID**

documentación	interno

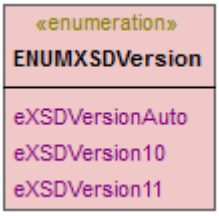
LiteralDeEnumeración **XMLValidator::ENUMXMLValidationMode::eProcessingModeValid**

documentación	Establece el modo de procesamiento XML 'validación'.

LiteralDeEnumeración **XMLValidator::ENUMXMLValidationMode::eProcessingModeWF**

documentación	Establece el modo de procesamiento XML 'formato correcto'. Este es el valor predeterminado

Enumeración **XMLValidator::ENUMXSDVersion**

diagrama	
elementosConTipo	Interfaz **XMLValidator** Interfaz **XQuery** Interfaz **XSLT** Operación **setXSDVersion** Operación **setXSDVersion** Operación **setXSDVersion**
documentación	Contiene los literales de enumeración que definen la versión del esquema XML que se usará para validar el documento.

LiteralDeEnumeración **XMLValidator::ENUMXSDVersion::eXSDVersion10**

documentación	El documento se validará con la versión XML-Schema 1.0.

LiteralDeEnumeración **XMLValidator::ENUMXSDVersion::eXSDVersion11**

documentación	El documento se validará con la versión XML-Schema 1.1.

 © 2013 Altova GmbH

LiteralDeEnumeración **XMLValidator::ENUMXSDVersion::eXSDVersionAuto**

documentación	Establece la versión de validación "detección automática".

Operación **XMLValidator::getLastErrorMessage**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	return	**return**	**String**		
documentación	Recupera el último mensaje de error del motor del validador XML. Parámetros: ninguno. Devuelve el texto del último mensaje de error, de tipo string.				

Operación **XMLValidator::isValid**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	tipo	**in**	**ENUMValidationType**		
	return	**return**	**boolean**		
documentación	Resultado de la validación XML con el esquema XML especificado con el valor de ENUMValidationType. Parámetros: 'type' contiene el valor del literal de enumeración. Genera una excepción RaptorXMLException cuando se produce un error. Si se produce un error durante la ejecución, use la operación getLastErrorMessage para obtener información adicional.				

Operación **XMLValidator::isValid**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	return	**return**	**boolean**		
documentación	Resultado de la validación XML. Parámetros: ninguno. Valores: 'true' si la operación finaliza correctamente, 'false' si falla.				

Operación **XMLValidator::isWellFormed**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	tipo	**in**	**ENUMWellformedCheckType**		
	return	**return**	**boolean**		

documentación	Resultado de la comprobación de formato XML especificado con el valor de ENUMWellformedCheckType. Parámetros: ninguno. Valores: 'true' si la operación finaliza correctamente, 'false' si falla. Genera una excepción RaptorXMLException cuando se produce un error. Si se produce un error durante la ejecución, use la operación getLastErrorMessage para obtener información adicional.

Operación **XMLValidator::isWellFormed**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	return	**return**	**boolean**		
documentación	Resultado de la comprobación de formato XML. Parámetros: ninguno. Valores: 'true' si la operación finaliza correctamente, 'false' si falla.				

Operación **XMLValidator::setSchemaImports**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	opt	**in**	**ENUMSchemaImports**		
	return	**return**	**void**		
documentación	Establece el comportamiento de los elementos xs:import y cada uno tiene un atributo de espacio de nombres opcional y un atributo schemaLocation opcional. Parámetros: 'opt' contiene el valor del literal de enumeración seleccionado ENUMSchemaImports.				

Operación **XMLValidator::setAssessmentMode**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	mode	**in**	**ENUMAssessmentMode**		
	return	**return**	**void**		
documentación	Establece el modo de evaluación del validador XML (Strict/Lax/Skip), depende de los literales ENUMAssessmentMode. Parámetros: 'mode' contiene el valor del literal de enumeración seleccionado.				

Operación **XMLValidator::setDTDFileName**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	filePath	**in**	**String**		
	return	**return**	**void**		

© 2013 Altova GmbH

documentación	Establece el documento DTD externo que se usará para la validación Parámetros: 'filePath' es una URL absoluta que da la ubicación de base de la DTD que se usará, de tipo string. Una URL absoluta especifica la ubicación exacta del archivo p. ej. http://www.miSitioWeb.com/archivosxml/miEntradaxml.xml.

Operación **XMLValidator::setDTDFromText**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	dtdText	**in**	**String**		
	return	**return**	**void**		
documentación	Establece el valor de texto para la DTD externa. Parámetros: 'dtdText' contiene la DTD como texto, de tipo string.				

Operación **XMLValidator::setEnableNamespaces**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	enable	**in**	**boolean**		
	return	**return**	**void**		
documentación	Habilita el procesamiento preparado para espacios de nombres. Es práctico para buscar errores en la instancia XML debidos a espacios de nombres incorrectos. Parámetros: 'enable' contiene el valor booleano, de tipo boolean. Valores: 'true' habilita el procesamiento preparado para espacios de nombres, 'false' lo deshabilita.				

Operación **XMLValidator::setInputXMLFileCollection**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	fileCollection	**in**			
	return	**return**	**void**		
documentación	Establece la colección de archivos XML que se usará como datos de entrada. Parámetros: 'fileCollection' es una colección de cadenas que contiene las URL absolutas de cada uno de los archivos XML.				

Operación **XMLValidator::setInputXMLFileName**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	filePath	**in**	**String**		
	return	**return**	**void**		

documentación	Establece el nombre de archivo (como URL) del archivo XML de entrada. Parámetros: 'filePath' es una URL absoluta que da la ubicación base del archivo XML, de tipo string. Una URL absoluta especifica la ubicación exacta del archivo p. ej. http://www.miSitioWeb.com/archivosxml/miEntradaxml.xml.

Operación **XMLValidator::setInputXMLFromText**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	inputText	**in**	**String**		
	return	**return**	**void**		
documentación	Suministra el contenido del documento XML de entrada como texto. Parámetros: 'inputText' contiene los datos XML, de tipo string. Nota: use la URL absoluta para el nombre de archivo de entrada. Una URL absoluta especifica la ubicación exacta del archivo p. ej. http://www.miSitioWeb.com/archivosxml/miEntradaxml.xml				

Operación **XMLValidator::setInputXMLTextCollection**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	stringCollection	**in**			
	return	**return**	**void**		
documentación	Establece la colección de archivos XML que se usará como datos de entrada. Parámetros: 'stringCollection' es una colección de cadenas que contiene los nombres de los documentos de entrada, sin tipo.				

Operación **XMLValidator::setPythonScriptFile**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	file	**in**	**String**		
	return	**return**	**void**		
documentación	Establece el nombre de archivo (como URL) del archivo de script Python. Parámetros: 'file' es una URL absoluta que da la ubicación base del archivo, de tipo string.				

Operación **XMLValidator::setSchemaFileCollection**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	fileCollection	**in**			
	return	**return**	**void**		

© 2013 Altova GmbH

documentación	Establece la colección de archivos que se usarán como esquemas XML externos. Parámetros: 'fileCollection' es una colección/matriz de URL absolutas que contiene la ubicación de los esquemas, sin tipo.

Operación **XMLValidator::setSchemaFileName**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	filePath	**in**	**String**		
	return	**return**	**void**		
documentación	Establece el nombre de archivo para el esquema XML externo. Parámetros: 'filePath' es la URL absoluta de la ubicación base del esquema, de tipo string.				

Operación **XMLValidator::setSchemaFromText**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	schemaText	**in**	**String**		
	return	**return**	**void**		
documentación	Suministra el contenido del esquema XML externo como texto. Parámetros: 'schemaText' es la cadena que contiene el esquema XML como texto.				

Operación **XMLValidator::setSchemalocationHints**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	opt	**in**	**ENUMLoadSchemalocation**		
	return	**return**	**void**		
documentación	Establece la ubicación desde la que se cargará el esquema, depiende de ENUMLoadSchemaLocation. Parámetros: 'opt' contiene el valor del literal de enumeración seleccionado, de tipo ENUMLoadSchemaLocation.				

Operación **XMLValidator::setSchemaMapping**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	opt	**in**	**ENUMSchemaMapping**		
	return	**return**	**void**		
documentación	Establece la ubicación desde la que se cargará el esquema, depende de ENUMSchemaMapping. Parámetros: 'opt' contiene el valor del literal de enumeración seleccionado, de tipo ENUMSchemaMapping.				

Operación **XMLValidator::setSchemaTextCollection**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	stringCollection	**in**			
	return	**return**	**void**		
documentación	Establece la colección de cadenas que usará como esquemas XML externos. Parámetros: 'stringCollection' es una colección/matriz de URL absolutas que contiene la ubicación de los esquemas, sin tipo.				

Operación **XMLValidator::setStreaming**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	support	**in**	**boolean**		
	return	**return**	**void**		
documentación	Habilita la validación de transmisión por secuencias. En este modo de validación, los datos almacenados en memoria se minimizan y se procesan más rápido. Parámetros: 'support' contiene el valor booleano, de tipo boolean. Valores: 'true' habilita la validación de transmisión por secuencias. 'false' la deshabilita. El valor predeterminado es true.				

Operación **XMLValidator::setXincludeSupport**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	support	**in**	**boolean**		
	return	**return**	**void**		
documentación	Habilita o deshabilita el uso de inclusiones XML (elementos XInclude). Parámetros: 'support' contiene el valor booleano, de tipo boolean. Valores: 'true' habilita los elementos XInclude include, 'false' los ignora.				

Operación **XMLValidator::setXMLValidationMode**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	mode	**in**	**ENUMXMLValidationMode**		
	return	**return**	**void**		
documentación	Establece el modo de validación XML, depende de ENUMXMLValidationMode. Parámetros: 'mode' contiene el valor del literal de enumeración seleccionado, de tipo ENUMXMLValidationMode.				

Operación **XMLValidator::setXSDVersion**

 © 2013 Altova GmbH

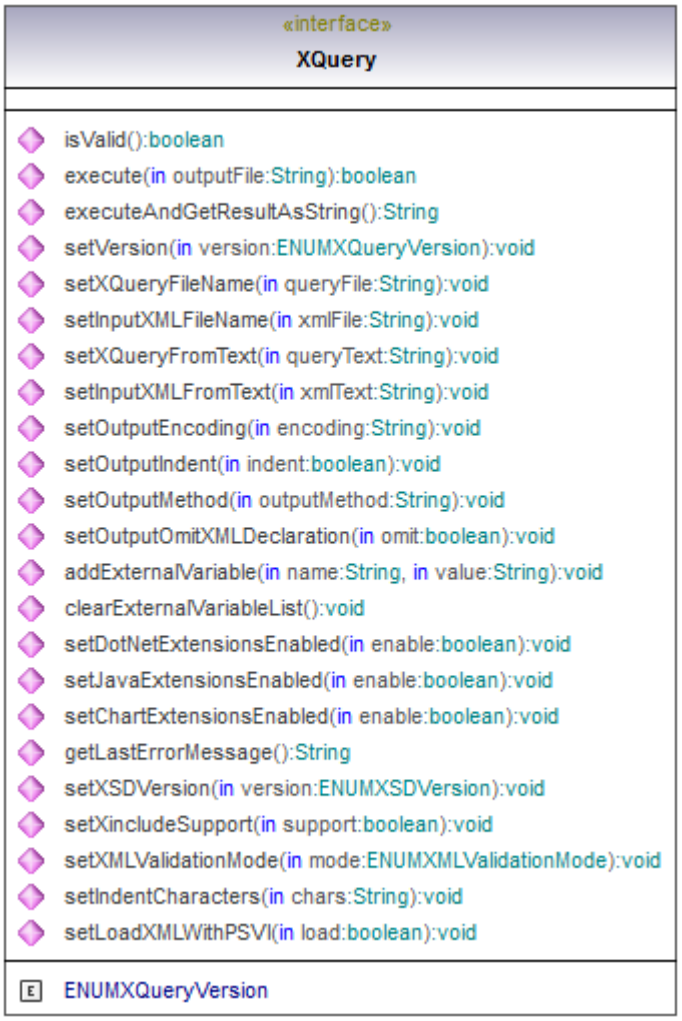
parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	version	**in**	**ENUMXSDVersion**		
	return	**return**	**void**		
documentación	Define la versión de XML Schema con la que se validará el documento. Parámetros: 'version' contiene la versión de XML Schema establecida por el literal EnumXSDVersion, de tipo EnumXSDVersion.				

Documentación UML generada con el editor UML **UModel**. **Más información en http://www.altova.com/es/umodel** 06/07/13 15:46:38

6.4 RaptorXMLJava - XQuery

Interfaz **XQuery**

diagrama



elementosConTipo Interfaz **RaptorXMLFactory** Operación **getXQuery**

© 2013 Altova GmbH

documentación	Ejecuta documentos XQuery 1.0 y 3.0 usando el motor RaptorXML. Los documentos XQuery y XML se pueden suministrar en forma de archivo (por medio de una URL) o, en la interfaz COM, como cadena de texto. El resultado se devuelve en forma de archivo (en la ubicación especificada) o, en la interfaz COM, como cadena de texto. Las variables XQuery externas se pueden suministrar a través de la línea de comandos y de la interfaz COM. Las opciones de serialización disponibles son: codificación de salida, método de salida (es decir, si el resultado es en formato XML, XHTML, HTML o texto), omisión de la declaración XML y sangría. Cuando las cadenas de entrada deban interpretarse como URL, se usarán rutas de acceso absolutas.

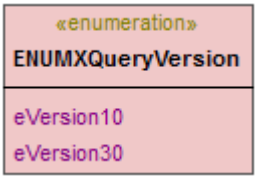
Operación **XQuery::addExternalVariable**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	nombre	**in**	**String**		
	value	**in**	**String**		
	return	**return**	**void**		
documentación	Añade el nombre y el valor de una variable externa nueva. Parámetros: 'name' es el nombre de la variable y es un QName válido, de tipo string. Cada variable externa y su valor se debe especificar en una llamada al método distinta. Las variables se deben declarar en el documento XQuery, con una declaración de tipo o no. Sea cual sea la declaración de tipo para la variable externa del documento XQuery, el valor de la variable suministrado al método AddExternalVariable no necesita delimitadores especiales, como las comillas, por ejemplo.				

Operación **XQuery::clearExternalVariableList**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	return	**return**	**void**		
documentación	Borra la lista de variables externas creadas con el método AddExternalVariable.				

Enumeración **XQuery::ENUMXQueryVersion**

diagrama	
elementosConTipo	Interfaz **XQuery** Operación **setVersion**

documentación	Contiene literales de enumeración que definen las versiones XQuery que se usarán: XQuery 1.0/3.0.

LiteralDeEnumeración **XQuery::ENUMXQueryVersion::eVersion10**

documentación	Establece que se usará la versión XQuery 1.0.

LiteralDeEnumeración **XQuery::ENUMXQueryVersion::eVersion30**

documentación	Establece que se usará la versión XQuery 3.0.

Operación **XQuery::execute**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	outputFile	**in**	**String**		
	return	**return**	**boolean**		
documentación	Ejecuta la transformación XQuery (dependiendo del valor de ENUMXQueryVersion) y guarda el resultado en un archivo de salida. Parámetros: 'outputFile' es la ruta (y nombre de archivo) del archivo de resultados. Si se produce un error durante la ejecución, use la operación getLastErrorMessage para obtener información adicional. Genera una excepción RaptorXMLException cuando se produce un error.				

Operación **XQuery::executeAndGetResultAsString**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	return	**return**	**String**		
documentación	Ejecuta el XQuery y devuelve el resultado en forma de cadena de texto. Parámetros: ninguno. Si ocurre un error durante la ejecución, use getLastErrorMessage para obtener información adicional. Genera una excepción RaptorXMLException cuando se produce un error.				

Operación **XQuery::getLastErrorMessage**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	return	**return**	**String**		
documentación	Recupera el último mensaje de error del motor XQuery. Parámetros: ninguno. Devuelve el texto del último mensaje de error, de tipo string.				

Operación **XQuery::isValid**

 © 2013 Altova GmbH

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	return	**return**	**boolean**		
documentación	Resultado de la validación XQuery 1.0/3.0 (dependiendo del valor de ENUMXQueryVersion). Parámetros: ninguno. Valores: 'true' si la operación funciona correctamente, 'false' si falla. Genera una excepción RaptorXMLException cuando se produce un error.				

Operación **XQuery::setChartExtensionsEnabled**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	enable	**in**	**boolean**		
	return	**return**	**void**		
documentación	Habilita o deshabilita las extensiones para gráficos de Altova. Parámetros: 'enable' contiene el valor booleano, de tipo boolean. Valores: 'true' habilita las extensiones para gráficos, 'false' las deshabilita.				

Operación **XQuery::setDotNetExtensionsEnabled**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	enable	**in**	**boolean**		
	return	**return**	**void**		
documentación	Habilita o deshabilita las extensiones para Visual Studio .NET. Parámetros: 'enable' contiene el valor booleano, de tipo boolean. Valores: 'true' habilita las extensiones .NET, 'false' las deshabilita.				

Operación **XQuery::setIndentCharacters**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	chars	**in**	**String**		
	return	**return**	**void**		
documentación	Establece la cadena de caracteres que se usará como sangría. Parámetros: 'chars' contiene el carácter de sangría, de tipo string.				

Operación **XQuery::setInputXMLFileName**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	xmlFile	**in**	**String**		
	return	**return**	**void**		

documentación	Establece el nombre de archivo (como URL) de la instancia XML de entrada que se debe transformar. Parámetros: 'xmlFile' contiene el nombre / la URL del archivo XML, de tipo string. Nota: use la URL absoluta para el nombre de archivo de entrada. Una URL absoluta especifica la ubicación exacta del archivo p. ej. http://www.miSitioWeb.com/archivosxml/miEntradaxml.xml

Operación **XQuery::setInputXMLFromText**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	xmlText	**in**	**String**		
	return	**return**	**void**		
documentación	Suministra el contenido del documento XML de entrada en forma de texto. Parámetros: 'xmlText' contiene los datos XML de entrada, de tipo string. Nota: use la URL absoluta para el nombre de archivo de entrada. Una URL absoluta especifica la ubicación exacta del archivo p. ej. http://www.miSitioWeb.com/archivosxml/miEntrada.xml				

Operación **XQuery::setJavaExtensionsEnabled**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	enable	**in**	**boolean**		
	return	**return**	**void**		
documentación	Habilita o deshabilita las extensiones Java. Parámetros: 'enable' contiene el valor booleano, de tipo boolean. Valores: 'true' habilita las extensiones Java, 'false' las deshabilita.				

Operación **XQuery::setLoadXMLWithPSVI**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	load	**in**	**boolean**		
	return	**return**	**void**		
documentación	Habilita o deshabilita la opción "Cargar conjunto de información posterior a la validación con esquema". Devuelve el valor booleano Valores: 'true' habilita el conjunto de información posterior a la validación con esquema, 'false' lo deshabilita.				

Operación **XQuery::setOutputEncoding**

 © 2013 Altova GmbH

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	encoding	**in**	**String**		
	return	**return**	**void**		
documentación	Establece la codificación para el documento de resultados. Parámetros: 'encoding' es el nombre de la codificación (p. ej. : UTF-8, UTF-16, ASCII, 8859-1, 1252), de tipo string.				

Operación **XQuery::setOutputIndent**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	indent	**in**	**boolean**		
	return	**return**	**void**		
documentación	Habilita o deshabilita la opción de sangría para el documento de resultados. Parámetros: 'indent' contiene el valor booleano, de tipo boolean. Valores: 'true' habilita la sangría, 'false' la deshabilita.				

Operación **XQuery::setOutputMethod**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	outputMethod	**in**	**String**		
	return	**return**	**void**		
documentación	Establece el método de serialización para el documento de resultados. Parámetros: 'outputMethod' contiene el método de serialización, de tipo string. Valores: xml, xhtml, html, text.				

Operación **XQuery::setOutputOmitXMLDeclaration**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	omit	**in**	**boolean**		
	return	**return**	**void**		
documentación	Habilita o deshabilita la generación de la declaración XML para el documento de resultados. Parámetros: 'omit' contiene el valor booleano, de tipo boolean. Valores: 'true' omite la declaración XML, 'false' la incluye.				

Operación **XQuery::setVersion**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	version	**in**	**ENUMXQueryVersion**		
	return	**return**	**void**		
documentación	Establece la versión XQuery que se utilizará (XQuery 1.0/3.0). Parámetros: 'version' contiene el número de versión establecido por el literal de enumeración EnumXQueryVersion eVersion10 o eVersion30.				

Operación **XQuery::setXincludeSupport**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	support	**in**	**boolean**		
	return	**return**	**void**		
documentación	Habilita o deshabilita el uso de inclusiones XML (elementos XInclude). Devuelve un valor booleano. Valores: 'true' habilita los elementos XInclude include, 'false' los ignora.				

Operación **XQuery::setXMLValidationMode**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	mode	**in**	**ENUMXMLValidationMode**		
	return	**return**	**void**		
documentación	Establece el modo de validación XML, depende de ENUMXMLValidationMode. 'mode' contiene el valor del literal de enumeración seleccionado, de tipo ENUMXMLValidationMode.				

Operación **XQuery::setXQueryFileName**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	queryFile	**in**	**String**		
	return	**return**	**void**		
documentación	Establece el nombre de archivo del documento XQuery. Parámetros: 'queryFile' contiene la URL absoluta que da la ubicación base del archivo XQuery, de tipo string. Una URL absoluta especifica la ubicación exacta del archivo p. ej. http://www.miSitioWeb.com/archivosxml/miEntradaxml.xml				

Operación **XQuery::setXQueryFromText**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	queryText	**in**	**String**		
	return	**return**	**void**		

© 2013 Altova GmbH

documentación	Suministra el contenido de la instrucción XQuery en forma de texto. Parámetros: 'queryText' contiene la instrucción XQuery, de tipo string.

Operación **XQuery::setXSDVersion**

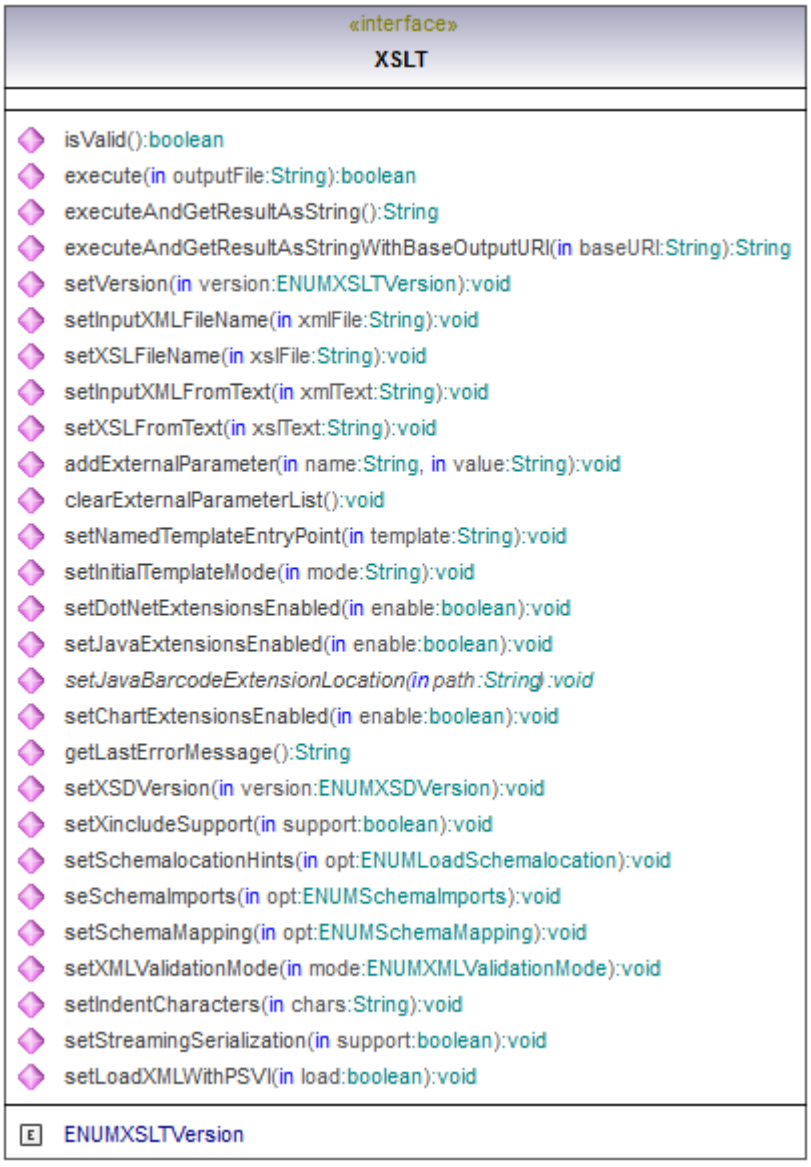
parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	version	**in**	**ENUMXSDVersion**		
	return	**return**	**void**		
documentación	Define la versión de XML Schema con la que se validará el documento. Parámetros: 'version' contiene la versión de XML Schema establecida por el literal EnumXSDVersion, de tipo EnumXSDVersion.				

© 2013 Altova GmbH

6.5 RaptorXMLJava - XSLT

Interfaz **XSLT**

diagrama



elementosConTipo Interfaz **RaptorXMLFactory** Operación **getXSLT**

© 2013 Altova GmbH

documentación	Transforma datos XML usando el documento XSLT 1.0, 2.0 o 3.0 suministrado. Los documentos XML y XSLT se pueden suministrar en forma de archivo (por medio de una URL) o, en el caso de la interfaz COM, en forma de cadena de texto. El resultado se devuelve en forma de archivo (en la ubicación especificada) o, en el caso de la interfaz COM, en forma de cadena de texto. Los parámetros XSLT se pueden suministrar por medio de la línea de comandos y por la interfaz COM. Las funciones de extensión de Altova habilitan un procesamiento especializado, como los gráficos, por ejemplo.

Operación **XSLT::addExternalParameter**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	nombre	**in**	**String**		
	value	**in**	**String**		
	return	**return**	**void**		

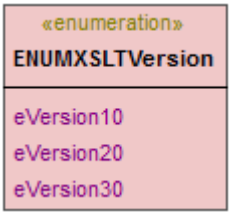
documentación	Añade el nombre y el valor de un parámetro externo nuevo. Parámetros: 'name' es el nombre de la variable y es un QName válido, de tipo string. 'value' es el valor de la variable, de tipo string. Como los valores del parámetro son expresiones XPath, los valores de parámetro que sean cadenas deben introducirse entre comillas simples.

Operación **XSLT::clearExternalParameterList**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	return	**return**	**void**		

documentación	Borra la lista de parámetros externos creada con el método AddExternalParameters. Parámetros: ninguno.

Enumeración **XSLT::ENUMXSLTVersion**

diagrama	
elementosConTipo	Interfaz **XSLT** Operación **setVersion**

documentación	Contiene los literales de enumeración que definen las versiones XSLT que se utilizarán: XSLT 1.0/2.0/3.0.

LiteralDeEnumeración **XSLT::ENUMXSLTVersion::eVersion10**

documentación	Establece que la versión XSLT que se utilizará es la versión 1.0.

LiteralDeEnumeración **XSLT::ENUMXSLTVersion::eVersion20**

documentación	Establece que la versión XSLT que se utilizará es la versión 2.0.

LiteralDeEnumeración **XSLT::ENUMXSLTVersion::eVersion30**

documentación	Establece que la versión XSLT que se utilizará es la versión 3.0.

Operación **XSLT::execute**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	outputFile	**in**	**String**		
	return	**return**	**boolean**		
documentación	Ejecuta la transformación XSLT (dependiendo del valor de ENUMXSLTVersion) y guarda el resultado en un archivo de salida. Parámetros: 'outputFile' es la ruta de acceso (y el nombre de archivo) del documento de salida, de tipo string. Si se produce un error durante la ejecución, use la operación getLastErrorMessage para obtener información adicional. Genera una excepción RaptorXMLException cuando se produce un error.				

Operación **XSLT::executeAndGetResultAsString**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	return	**return**	**String**		
documentación	Ejecuta el XSLT y devuelve el resultado en forma de cadena. Parámetros: ninguno. Si se produce un error durante la ejecución, use la operación getLastErrorMessage para obtener información adicional. Genera una excepción RaptorXMLException cuando se produce un error.				

Operación **XSLT::executeAndGetResultAsStringWithBaseOutputURI**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	baseURI	**in**	**String**		
	return	**return**	**String**		

© 2013 Altova GmbH

documentación	Ejecuta el XSLT y devuelve el resultado en forma de cadena en la ubicación definida por el URI base. Parámetros: ninguno. Si se produce un error durante la ejecución, use la operación getLastErrorMessage para obtener información adicional. Genera una excepción RaptorXMLException cuando se produce un error.

Operación **XSLT::getLastErrorMessage**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	return	**return**	**String**		
documentación	Recupera el último mensaje de error del motor XSLT. Parámetros: ninguno.				

Operación **XSLT::isValid**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	return	**return**	**boolean**		
documentación	Resultado de la validación XSLT (depende del valor de ENUMXSLTVersion) Parámetros: ninguno. Valores: 'true' si la operación funciona correctamente, 'false' si falla. Genera una excepción RaptorXMLException cuando se produce un error.				

Operación **XSLT::setSchemaImports**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	opt	**in**	**ENUMSchemaImports**		
	return	**return**	**void**		
documentación	Define el comportamiento de los elementos xs:import. Cada elemento tiene un atributo de espacio de nombres opcional y un atributo schemaLocation opcional, dependiendo de ENUMSchemaImports.				

Operación **XSLT::setChartExtensionsEnabled**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	enable	**in**	**boolean**		
	return	**return**	**void**		
documentación	Habilita o deshabilita las extensiones de Altova para gráficos. Parámetros: 'enable' contiene el valor booleano, de tipo boolean. Valores: 'true' habilita las extensiones para gráficos, 'false' las deshabilita.				

Operación **XSLT::setDotNetExtensionsEnabled**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	enable	**in**	**boolean**		
	return	**return**	**void**		
documentación	Habilita o deshabilita las extensiones para Visual Studio .NET. Parámetros: 'enable' contiene el valor booleano, de tipo boolean. Valores: 'true' habilita las extensiones para .NET, 'false' las deshabilita.				

Operación **XSLT::setIndentCharacters**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	chars	**in**	**String**		
	return	**return**	**void**		
documentación	Establece la cadena de caracteres que se usará como sangría. Parámetros: 'chars' es el carácter de sangría, de tipo string.				

Operación **XSLT::setInitialTemplateMode**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	mode	**in**	**String**		
	return	**return**	**void**		
documentación	Establece el modo inicial para el procesamiento XSLT. Parámetros: 'mode' es el nombre del modo inicial necesario, de tipo string. Se procesarán las plantillas que tengan este modo. Por ejemplo: SetInitialTemplateMode="MiModo". Nota: la transformación siempre debe producirse después de asignar los documentos XML y XSLT.				

Operación **XSLT::setInputXMLFileName**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	xmlFile	**in**	**String**		
	return	**return**	**void**		
documentación	Establece el nombre de archivo (en forma de URL) de la instancia XML de entrada que debe transformarse. Parámetros: 'xmlFile' contiene el nombre / la URL del archivo XML, de tipo string. Nota: use la URL absoluta para el nombre de archivo de entrada. Una URL absoluta especifica la ubicación exacta del archivo p. ej. http://www.miSitioWeb.com/archivosxml/miEntradaxml.xml				

 © 2013 Altova GmbH

Operación **XSLT::setInputXMLFromText**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	xmlText	**in**	**String**		
	return	**return**	**void**		
documentación	Suministra el contenido del documento XML de entrada en forma de texto. Parámetros: 'xmlText' contiene los datos XML de entrada, de tipo string. Nota: use la URL absoluta para el nombre de archivo de entrada. Una URL absoluta especifica la ubicación exacta del archivo p. ej. http://www.miSitioWeb.com/archivosxml/miEntradaxml.xml				

Operación **XSLT::setJavaBarcodeExtensionLocation**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	path	**in**	**String**		
	return	**return**	**void**		
documentación	Define la ubicación del archivo de extensión de código de barras Java. Parámetros: 'path' contiene la ubicación del archivo de extensión, de tipo string. Use la URL absoluta para el nombre de archivo de entrada. Una URL absoluta especifica la ubicación exacta del archivo p. ej. http://www.myWebsite.com/xmlfiles/miEntradaxml.xml				

Operación **XSLT::setJavaExtensionsEnabled**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	enable	**in**	**boolean**		
	return	**return**	**void**		
documentación	Habilita o deshabilita las extensiones Java. Parámetros: 'enable' contiene el valor booleano, de tipo boolean. Valores: 'true' habilita las extensiones Java, 'false' las deshabilita.				

Operación **XSLT::setLoadXMLWithPSVI**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	load	**in**	**boolean**		
	return	**return**	**void**		

documentación	Habilita o deshabilita la opción "Cargar conjunto de información posterior a la validación con esquema". Parámetros: 'load' contiene el valor booleano, de tipo boolean. Valores: 'true' habilita el conjunto de información posterior a la validación con esquema, 'false' lo deshabilita.

Operación **XSLT::setNamedTemplateEntryPoint**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	template	**in**	**String**		
	return	**return**	**void**		

documentación	Establece el punto de entrada de la plantilla con nombre. Parámetros: 'template' es el nombre del nodo desde el cual debe empezar el procesamiento, de tipo string.

Operación **XSLT::setSchemalocationHints**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	opt	**in**	**ENUMLoadSchemalocation**		
	return	**return**	**void**		

documentación	Establece la ubicación desde la que se cargará el esquema, depende de ENUMLoadSchemaLocation. Parámetros: 'opt' contiene el valor del literal de enumeración seleccionado, de tipo ENUMLoadSchemaLocation.

Operación **XSLT::setSchemaMapping**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	opt	**in**	**ENUMSchemaMapping**		
	return	**return**	**void**		

documentación	Establece la ubicación desde la que se cargará el esquema, depende de ENUMSchemaMapping. Parámetros: 'opt' contiene el valor del literal de enumeración seleccionado, de tipo ENUMSchemaMapping.

Operación **XSLT::setStreamingSerialization**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	support	**in**	**boolean**		
	return	**return**	**void**		

© 2013 Altova GmbH

documentación	Habilita la serialización de secuencias de datos. En el modo de transmisión por secuencias, los datos almacenados en memoria se minimizan y se procesan más rápido. Parámetros: 'support' contiene el valor booleano, de tipo boolean. Valores: 'true' habilita la serialización de secuencias de datos, 'false' la deshabilita. El valor predeterminado es 'true'.

Operación **XSLT::setVersion**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	version	**in**	**ENUMXSLTVersion**		
	return	**return**	**void**		
documentación	Establece la versión XQuery que se utilizará (XQuery 1.0/2.0/3.0). Parámetros: 'version' contiene el número de versión establecido por los literales de la enumeración EnumXSLTVersion, de tipo EnumXSLTVersion.				

Operación **XSLT::setXincludeSupport**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	support	**in**	**boolean**		
	return	**return**	**void**		
documentación	Habilita o deshabilita el uso de inclusiones XML (elementos XInclude). Parámetros: 'support' contiene el valor booleano, de tipo boolean. Valores: 'true' habilita los elementos XInclude include, 'false' los ignora.				

Operación **XSLT::setXMLValidationMode**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	mode	**in**	**ENUMXMLValidationMode**		
	return	**return**	**void**		
documentación	Establece el modo de validación XML, depende de ENUMXMLValidationMode. Parámetros: 'mode' contiene el valor del literal de enumeración seleccionado, de tipo ENUMXMLValidationMode.				

Operación **XSLT::setXSDVersion**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	version	**in**	**ENUMXSDVersion**		
	return	**return**	**void**		
documentación	Define la versión de XML Schema que se usará para validar el documento. Parámetros: 'version' contiene la versión de XML Schema establecida por el literal EnumXSDVersion, de tipo EnumXSDVersion.				

Operación **XSLT::setXSLFileName**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	xslFile	**in**	**String**		
	return	**return**	**void**		
documentación	Establece la ruta/URL necesaria para ubicar el archivo XSLT que se debe usar para la transformación. Parámetros: 'xslFile' es la ruta de archivo / el nombre de archivo del archivo XSLT, de tipo string. Nota: use la URL absoluta para el nombre de archivo de entrada. Una URL absoluta especifica la ubicación exacta del archivo p. ej. http://www.miSitioWeb.com/archivosxml/miEntradaxml.xml.				

Operación **XSLT::setXSLFromText**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	xslText	**in**	**String**		
	return	**return**	**void**		
documentación	Establece el nombre del archivo XSLT que se debe usar para la transformación. Parámetros: 'xslText' es el nombre de archivo de texto XML del archivo XSLT, de tipo string.				

Documentación UML generada con el editor UML **UModel**. **Más información en http://www.altova.com/es/umodel** 06/07/13 15:46:38

6.6 RaptorXMLJava - RaptorXMLException

Operación **RaptorXMLException::RaptorXMLException**

parámetro	nombre	dirección	tipo	multiplicidad	predeterminado
	message	**in**	**String**		

Documentación UML generada con el editor UML **UModel**. **Más información en http://www.altova.com/es/umodel** 06/07/13 15:46:38

Altova RaptorXML 2013

Interfaz .NET/COM

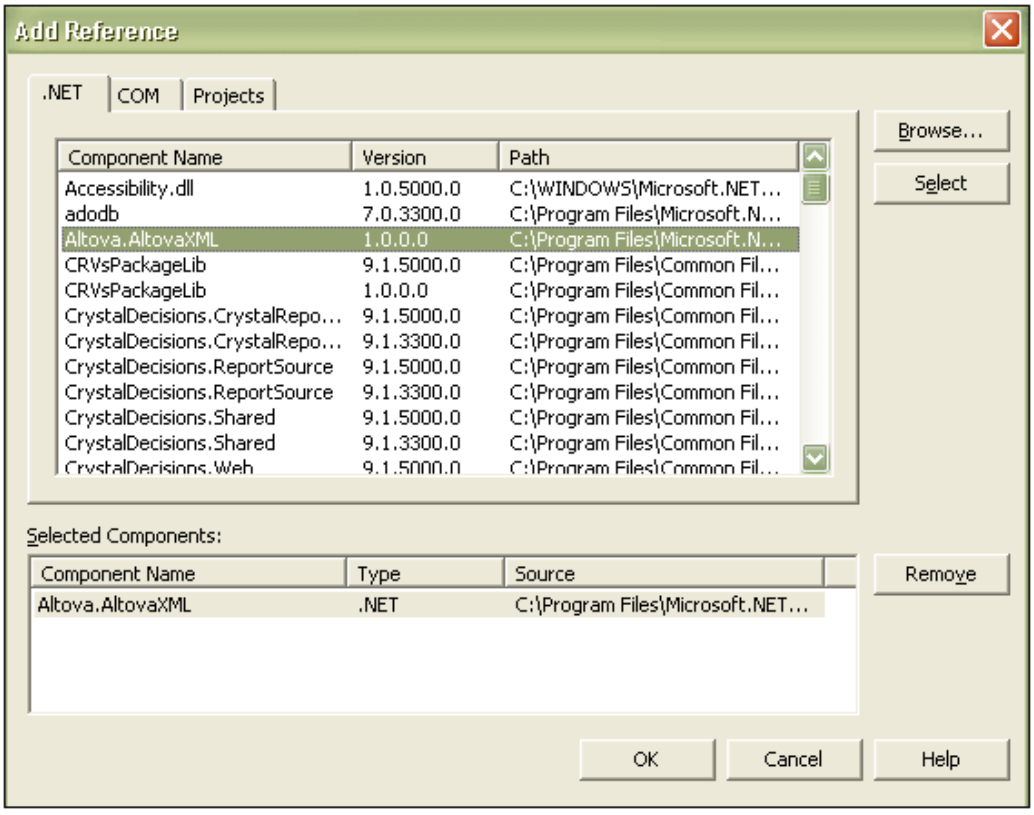
7 Interfaz .NET/COM

La interfaz .NET está construida como una clase contenedora alrededor de la interfaz COM de RaptorXML COM. Se ofrece como ensamblado de interoperabilidad principal firmado por Altova y usa el espacio de nombres `Altova.RaptorXML`. Para usar RaptorXML en su proyecto .NET, necesita: (i) añadir en su proyecto una referencia al DLL de RaptorXML (que se llama `Altova.RaptorXML.dll`) y (ii) tener RaptorXML registrado como objeto servidor COM. Después podrá usar las funciones de RaptorXML en su proyecto.

Añadir al proyecto una referencia al DLL de RaptorXML

El paquete de RaptorXML contiene un archivo DLL firmado y llamado `Altova.RaptorXML.dll`, que se añadirá automáticamente al caché global de ensamblados (y a la biblioteca de referencias .NET) cuando se instale RaptorXML usando el instalador 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), que enumera una lista de los componentes .NET instalados. (Nota: si el componente RaptorXML no está en la pestaña .NET, puede encontrarlo en la pestaña COM.)



2. Seleccione `Altova.RaptorXML` en la lista de componentes, haga doble clic en él (o pulse el botón **Seleccionar**) y después haga clic en **Aceptar**

Registrar RaptorXML como objeto servidor COM

Este registro lo hace el instalador de RaptorXML automáticamente. Si cambia la ubicación del archivo `RaptorXML_COM.exe` tras la instalación, debería registrar RaptorXML como objeto

 © 2013 Altova GmbH

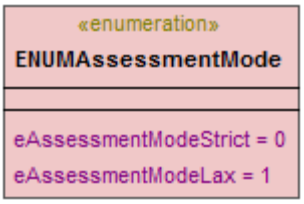
servidor COM ejecutando el comando `RaptorXML_COM.exe /regserver`. (Recuerde que debe introducir la ruta correcta de `RaptorXML_COM.exe`. Consulte el apartado Registrar RaptorXML como objeto servidor COM para obtener más información.)

Cuando `Altova.RaptorXML.dll` ya esté a disposición de la interfaz .NET y RaptorXML esté registrado como objeto servidor COM, las funciones de RaptorXML estarán disponibles en su proyecto .NET.

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.

7.1 RaptorXMLDev_COM - ENUMAssessmentMode

Enumeración **ENUMAssessmentMode**

diagrama	
elementosConTipo	Interfaz **IXMLValidator** Operación **AssessmentMode**
documentación	Contiene los literales de enumeración que definen el modo de evaluación del validador XML: Strict/Lax.

LiteralDeEnumeración **ENUMAssessmentMode::eAssessmentModeLax**

documentación	Establece el modo de evaluación de validez del esquema lax.

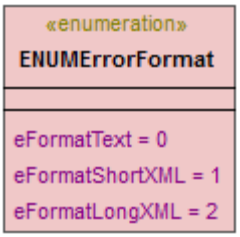
LiteralDeEnumeración **ENUMAssessmentMode::eAssessmentModeStrict**

documentación	Establece el modo de evaluación de validez del esquema strict.

Documentación UML generada con el editor UML **UModel**. **Más información en http://www.altova.com/es/umodel** 06/07/13 15:48:10

7.2 RaptorXMLDev_COM - ENUMErrorFormat

Enumeración **ENUMErrorFormat**

diagrama	
elementosConTipo	Interfaz **IApplication** Operación **ErrorFormat**
documentación	Contiene los literales de enumeración que definen el formato de los errores de salida.

LiteralDeEnumeración **ENUMErrorFormat::eFormatLongXML**

documentación	Establece el formato de errores de salida "XML largo". Este formato XML de salida es el que más detalles ofrece.

LiteralDeEnumeración **ENUMErrorFormat::eFormatShortXML**

documentación	Establece el formato de errores de salida "XML breve". Este formato XML de salida ofrece información más resumida que el formato "XML largo".

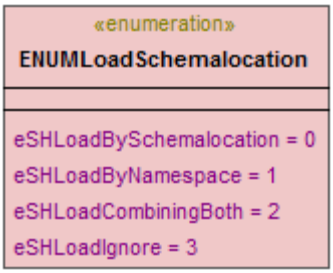
LiteralDeEnumeración **ENUMErrorFormat::eFormatText**

documentación	Establece el formato de errores de salida "texto". Esta es la opción predeterminada.

Documentación UML generada con el editor UML **UModel**. **Más información en http://www.altova.com/es/umodel** 06/07/13 15:48:10

7.3 RaptorXMLDev_COM - ENUMLoadSchemalocation

Enumeración **ENUMLoadSchemalocation**

diagrama	
elementosConTipo	Interfaz **IXBRL** Operación **SchemalocationHints** Interfaz **IXMLValidator** Operación **SchemalocationHints** Interfaz **IXSLT** Operación **SchemalocationHints**
documentación	Contiene los literales de enumeración que definen el comportamiento de LoadSchemaLocation y cada uno tiene un atributo de espacio de nombres opcional y un atributo schemaLocation opcional.

LiteralDeEnumeración **ENUMLoadSchemalocation::eSHLoadByNamespace**

documentación	Asigna LoadByNamespace a LoadSchemaLocation. Usa la parte de espacio de nombres de xsi:schemaLocation y una cadena vacía en el caso de xsi:noNamespaceSchemaLocation y ubica el esquema por medio de una asignación de catálogo.

LiteralDeEnumeración **ENUMLoadSchemalocation::eSHLoadBySchemalocation**

documentación	Asigna LoadBySchemalocation a LoadSchemaLocation. Usa la URL de la ubicación del esquema de los atributos xsi:schemaLocation y xsi:noNamespaceSchemaLocation en documentos de instancia XML o XBRL. Este es el valor predeterminado.

LiteralDeEnumeración **ENUMLoadSchemalocation::eSHLoadCombiningBoth**

documentación	Asigna CombiningBoth a LoadSchemaLocation. Si el espacio de nombres o la URL tiene una asignación de catálogo, se utiliza la asignación. Si ambos tienen una asignación de catálogo, el valor del parámetro schema-mapping decide qué asignación se utiliza. Si ninguno de los dos tiene una asignación de catálogo, se utiliza la URL.

LiteralDeEnumeración **ENUMLoadSchemalocation::eSHLoadIgnore**

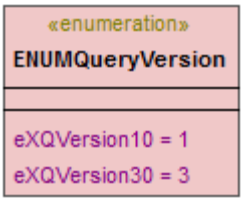
documentación	Asigna LoadIgnore a LoadSchemaLocation. Si el valor del parámetro es 'ignore', se ignoran los atributos xsi:schemaLocation y xsi:noNamespaceSchemaLocation.

 © 2013 Altova GmbH

Documentación UML generada con el editor UML **UModel**. **Más información en http://www.altova.com/es/umodel** 06/07/13 15:48:10

7.4 RaptorXMLDev_COM - ENUMQueryVersion

Enumeración **ENUMQueryVersion**

diagrama	
documentación	Contiene los literales de enumeración que definen las versiones XQuery que se utilizarán: XQuery 1.0/3.0.

LiteralDeEnumeración **ENUMQueryVersion::eXQVersion10**

documentación	Establece que se usará la versión XQuery 1.0.

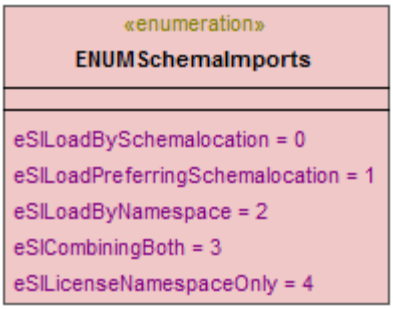
LiteralDeEnumeración **ENUMQueryVersion::eXQVersion30**

documentación	Establece que se usará la versión XQuery 3.0.

Documentación UML generada con el editor UML **UModel**. **Más información en http://www.altova.com/es/umodel** 06/07/13 15:48:10

7.5 RaptorXMLDev_COM - ENUMSchemaImports

Enumeración **ENUMSchemaImports**

diagrama	
elementosConTipo	Interfaz **IXBRL** Operación **SchemaImports** Interfaz **IXMLValidator** Operación **SchemaImports** Interfaz **IXSLT** Operación **SchemaImports**
documentación	Contiene los literales de enumeración que definen el comportamiento de los elementos xs:import y cada uno tiene un atributo de espacio de nombres especial y un atributo schemaLocation opcional.

LiteralDeEnumeración **ENUMSchemaImports::eSICombiningBoth**

documentación	Asigna CombiningBoth a SchemaImport. Si el espacio de nombres o la URL tiene una asignación de catálogo, se utiliza la asignación. Si ambos tienen una asignación de catálogo, el valor del parámetro schema-mapping decide qué asignación se utiliza. Si ni el espacio de nombres ni la URL tienen una asignación de catálogo, se usa la URL.

LiteralDeEnumeración **ENUMSchemaImports::eSILicenseNamespaceOnly**

documentación	Asigna LicenseNamespaceOnly a SchemaImport. Se importa el espacio de nombres. No se importa el documento de esquema.

LiteralDeEnumeración **ENUMSchemaImports::eSILoadByNamespace**

documentación	Asigna LoadByNamespace a SchemaImport. El valor del atributo de espacio de nombres se usa para ubicar el esquema por medio de una asignación de catálogo.

LiteralDeEnumeración **ENUMSchemaImports::eSILoadBySchemalocation**

documentación	Asigna LoadBySchemalocation a SchemaImport. El valor del atributo schemaLocation se usa para ubicar el esquema por medio de una asignación de catálogo. Si está presente el atributo de espacio de nombres, se importa el espacio de nombres (con licencia).

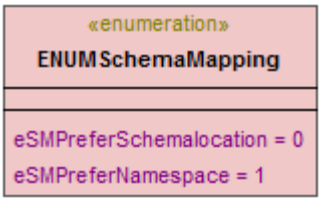
LiteralDeEnumeración **ENUMSchemaImports::eSILoadPreferringSchemalocation**

documentación	Asigna LoadPreferringSchemalocation a SchemaImport. Si está presente, se usa el atributo schemaLocation, teniendo en cuenta las asignaciones de catálogo. Si no está presente, se usa el valor del atributo de espacio de nombres por medio de una asignación de catálogo. Este es el valor predeterminado. Este es el valor predeterminado.

Documentación UML generada con el editor UML **UModel**. **Más información en http://www.altova.com/es/umodel** 06/07/13 15:48:10

7.6 RaptorXMLDev_COM - ENUMSchemaMapping

Enumeración **ENUMSchemaMapping**

diagrama	
elementosConTipo	Interfaz **IXBRL** — Operación **SchemaMapping** Interfaz **IXMLValidator** — Operación **SchemaMapping** Interfaz **IXSLT** — Operación **SchemaMapping**
documentación	Contiene los literales de enumeración que definen cuál de las dos asignaciones de catálogo se utilizan.

LiteralDeEnumeración **ENUMSchemaMapping::eSMPreferNamespace**

documentación	Asigna PreferNamespace a la ubicación de la asignación de esquema.

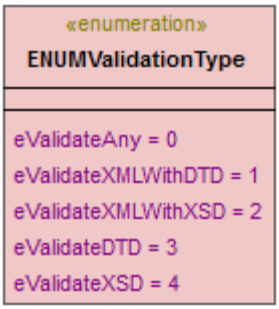
LiteralDeEnumeración **ENUMSchemaMapping::eSMPreferSchemalocation**

documentación	Asigna PreferSchemaLocation a la ubicación de la asignación de esquema. El valor de PreferSchemaLocation remite a la asignación de URL.

Documentación UML generada con el editor UML **UModel**. **Más información en http://www.altova.com/es/umodel** 06/07/13 15:48:10

7.7 RaptorXMLDev_COM - ENUMValidationType

Enumeración **ENUMValidationType**

diagrama	
elementosConTipo	Interfaz **IXMLValidator** Operación **IsValid**
documentación	Contiene los literales de enumeración que definen el tipo de documento que se debe validar.

LiteralDeEnumeración **ENUMValidationType::eValidateAny**

documentación	Establece el tipo de validación 'any'. Esto valida cualquier tipo de documento. El tipo de documento se detecta automáticamente.

LiteralDeEnumeración **ENUMValidationType::eValidateDTD**

documentación	Establece el tipo de validación 'validate DTD'. Esto valida un documento DTD.

LiteralDeEnumeración **ENUMValidationType::eValidateXMLWithDTD**

documentación	Establece el tipo de validación 'XML with DTD'. Esto valida un documento XML con una DTD.

LiteralDeEnumeración **ENUMValidationType::eValidateXMLWithXSD**

documentación	Establece el tipo de validación 'XML with XSD'. Esto valida un documento XML con un esquema XML.

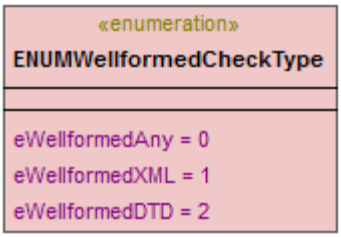
LiteralDeEnumeración **ENUMValidationType::eValidateXSD**

documentación	Establece el tipo de validación 'validate XSD'. Esto valida un esquema XML del W3C.

 © 2013 Altova GmbH

7.8 RaptorXMLDev_COM - ENUMWellformedCheckType

Enumeración **ENUMWellformedCheckType**

diagrama	
elementosConTipo	Interfaz **IXMLValidator** Operación **IsWellFormed**
documentación	Contiene los literales de enumeración que definen el tipo de comprobación de formato que se debe realizar.

LiteralDeEnumeración **ENUMWellformedCheckType::eWellformedAny**

documentación	Establece el tipo de comprobación de formato 'Any' Comprueba si el documento XML o DTD tiene un formato correcto, según la especificación correspondiente. El tipo de documento se detecta automáticamente.

LiteralDeEnumeración **ENUMWellformedCheckType::eWellformedDTD**

documentación	Establece el tipo de comprobación de formato 'DTD' Comprueba si los documentos DTD tienen un formato correcto según la especificación XML 1.0 o XML 1.1. Con RaptorXML Server y RaptorXML+XBRL Server puede realizar comprobaciones de formato en varios documentos a la vez.

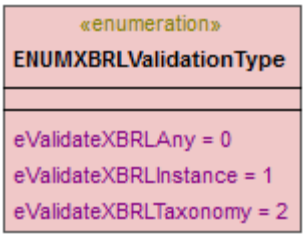
LiteralDeEnumeración **ENUMWellformedCheckType::eWellformedXML**

documentación	Establece el tipo de comprobación de formato 'XML' Comprueba si los documentos XML tienen un formato correcto según la especificación XML 1.0 o XML 1.1. Con RaptorXML Server y RaptorXML+XBRL Server puede realizar comprobaciones de formato en varios documentos a la vez.

Documentación UML generada con el editor UML **UModel**. **Más información en http://www.altova.com/es/umodel** 06/07/13 15:48:10

7.9 RaptorXMLDev_COM - ENUMXBRLValidationType

Enumeración **ENUMXBRLValidationType**

diagrama	
elementosConTipo	Interfaz **IXBRL** Operación **IsValid**
documentación	Contiene los literales de enumeración que definen el tipo de documento XBRL que se debe validar. Función disponible solamente en RaptorXML+XBRL

LiteralDeEnumeración **ENUMXBRLValidationType::eValidateXBRLAny**

documentación	Establece el tipo de validación 'any'. Esto valida cualquier documento XBRL. El tipo de documento se detecta automáticamente.

LiteralDeEnumeración **ENUMXBRLValidationType::eValidateXBRLInstance**

documentación	Establece el tipo de validación 'instance'. Esto valida un documento de instancia XBRL.

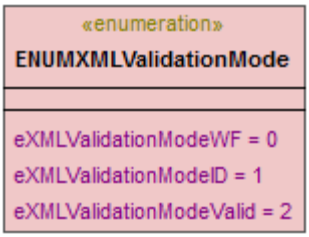
LiteralDeEnumeración **ENUMXBRLValidationType::eValidateXBRLTaxonomy**

documentación	Establece el tipo de validación 'taxonomy'. Esto valida un documento de taxonomía XBRL.

Documentación UML generada con el editor UML **UModel**. **Más información en http://www.altova.com/es/umodel** 06/07/13 15:48:10

7.10 RaptorXMLDev_COM - ENUMXMLValidationMode

Enumeración **ENUMXMLValidationMode**

diagrama	
elementosConTipo	Interfaz **IXMLValidator** Operación **XMLValidationMode** Interfaz **IXQuery** Operación **XMLValidationMode** Interfaz **IXSLT** Operación **XMLValidationMode**
documentación	Contiene los literales de enumeración que definen el modo de procesamiento que se utilizará.

LiteralDeEnumeración **ENUMXMLValidationMode::eXMLValidationModeID**

documentación	de uso interno

LiteralDeEnumeración **ENUMXMLValidationMode::eXMLValidationModeValid**

documentación	Establece el modo de procesamiento XML 'validation'.

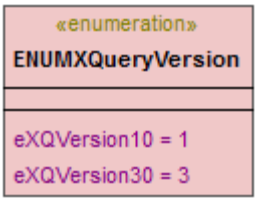
LiteralDeEnumeración **ENUMXMLValidationMode::eXMLValidationModeWF**

documentación	Establece el modo de procesamiento XML 'well-formed'. Este es el valor predeterminado.

Documentación UML generada con el editor UML **UModel**. **Más información en http://www.altova.com/es/umodel** 06/07/13 15:48:10

7.11 RaptorXMLDev_COM - ENUMXQueryVersion

Enumeración **ENUMXQueryVersion**

diagrama	
elementosConTipo	Interfaz **IXQuery** Operación **EngineVersion**
documentación	Contiene los literales de enumeración que definen las versiones XQuery que se van a utilizar: XQuery 1.0/3.0.

LiteralDeEnumeración **ENUMXQueryVersion::eXQVersion10**

documentación	Establece que la versión que se utilizará es XQuery 1.0.

LiteralDeEnumeración **ENUMXQueryVersion::eXQVersion30**

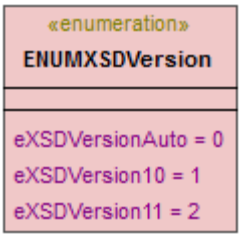
documentación	Establece que la versión que se utilizará es XQuery 3.0.

Documentación UML generada con el editor UML **UModel**. **Más información en http://www.altova.com/es/umodel** 06/07/13 15:48:10

© 2013 Altova GmbH

7.12 RaptorXMLDev_COM - ENUMXSDVersion

Enumeración **ENUMXSDVersion**

diagrama	
elementosConTipo	Interfaz **IXMLValidator** Operación **XSDVersion** Interfaz **IXQuery** Operación **XSDVersion** Interfaz **IXSLT** Operación **XSDVersion**
documentación	Contiene los literales de enumeración que definen la versión de XML Schema que se utilizará para validar el documento.

LiteralDeEnumeración **ENUMXSDVersion::eXSDVersion10**

documentación	Establece que la versión que se utilizará para validar el esquema es la versión XML-Schema 1.0.

LiteralDeEnumeración **ENUMXSDVersion::eXSDVersion11**

documentación	Establece que la versión que se utilizará para validar el esquema es la versión XML-Schema 1.1.

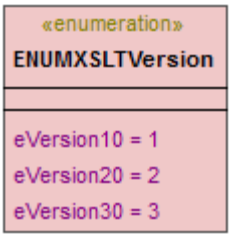
LiteralDeEnumeración **ENUMXSDVersion::eXSDVersionAuto**

documentación	Establece que la versión que se utilizará para validar el esquema es la versión 'auto-detect'.

Documentación UML generada con el editor UML **UModel**. **Más información en http://www.altova.com/es/umodel** 06/07/13 15:48:10

7.13 RaptorXMLDev_COM - ENUMXSLTVersion

Enumeración **ENUMXSLTVersion**

diagrama	
elementosConTipo	Interfaz **IXSLT** Operación **EngineVersion**
documentación	Contiene los literales de enumeración que definen las versiones XSLT que se usarán: XSLT 1.0/2.0/3.0.

LiteralDeEnumeración **ENUMXSLTVersion::eVersion10**

documentación	Establece que se usará la versión XSLT 1.0.

LiteralDeEnumeración **ENUMXSLTVersion::eVersion20**

documentación	Establece que se usará la versión XSLT 2.0.

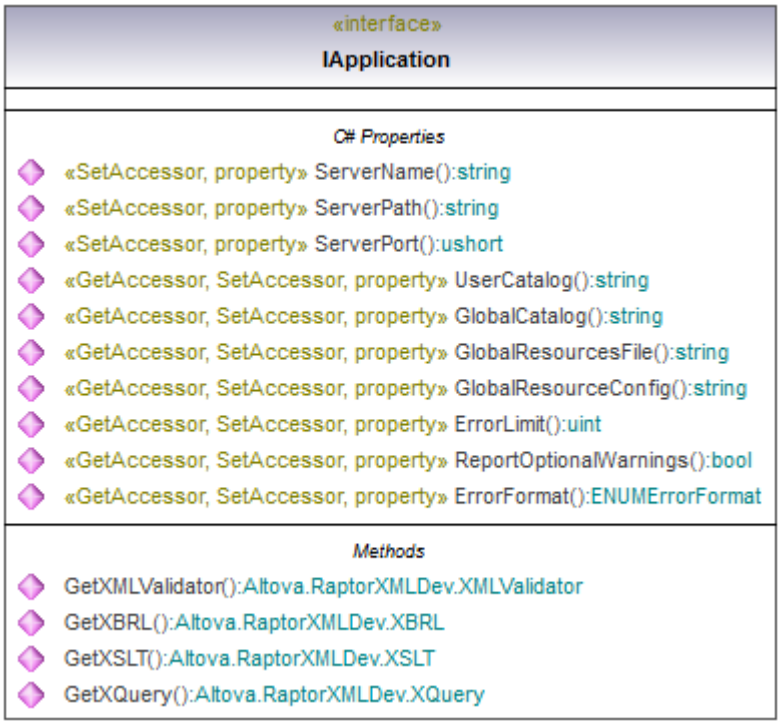
LiteralDeEnumeración **ENUMXSLTVersion::eVersion30**

documentación	Establece que se usará la versión XSLT 3.0.

Documentación UML generada con el editor UML **UModel**. **Más información en http://www.altova.com/es/umodel** 06/07/13 15:48:10

7.14 RaptorXMLDev_COM - IApplication

Interfaz **IApplication**

diagrama	
documentación	Descripción de la interfaz IApplication

Operación **IApplication::ErrorFormat**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**ENUMErrorFormat**			
documentación	Establece el formato de error de RaptorXML (Text, ShortXML, LongXML), dependiendo de los literales ENUMErrorFormat. Parámetros: Devuelve el valor del literal de enumeración seleccionado, de tipo ENUMErrorFormat.					

Operación **IApplication::ErrorLimit**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**uint**			
documentación	Configura la propiedad de límite de errores de validación de RaptorXML. Propiedades: Define el número de errores que se deben notificar antes de detener la ejecución, de tipo uint.					

Operación **IApplication::GetXBRL**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**Altova.RaptorXMLDev.XBRL**			
documentación	Recupera el motor XBRL y devuelve una instancia XBRL nueva de RaptorXMLFactory. Compatible con RaptorXML+XBRL Server solamente Propiedades: Devuelve una instancia del motor XBRL.					

Operación **IApplication::GetXMLValidator**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**Altova.RaptorXMLDev.XMLValidator**			
documentación	Recupera el motor XML y devuelve una instancia del validador XML nueva de RaptorXMLFactory. Propiedades: Devuelve una instancia del validador XML.					

Operación **IApplication::GetXQuery**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**Altova.RaptorXMLDev.XQuery**			
documentación	Recupera el motor XQuery y devuelve una instancia XQuery nueva de RaptorXMLFactory. Propiedades: Devuelve una instancia del motor XQuery.					

Operación **IApplication::GetXSLT**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**Altova.RaptorXMLDev.XSLT**			
documentación	Recupera el motor XSLT y devuelve una instancia XSLT nueva de XMLFactory. Propiedades: Devuelve una instancia del motor XSLT.					

© 2013 Altova GmbH

Operación **IApplication::GlobalCatalog**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**string**			
documentación	Establece le nombre de archivo (como URL) del catálogo global (p. ej. RootCatalog.xml). Propiedades: La URL para la ubicación base del archivo de catálogo global, de tipo string. Nota: use la URL absoluta para el nombre de archivo de entrada. Una URL absoluta especifica la ubicación exacta del archivo p. ej. http://www.miSitioWeb.com/archivosxml/miCatGlobal.xml					

Operación **IApplication::GlobalResourceConfig**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**string**			
documentación	Establece la configuración activa del recurso global. Propiedades: El nombre de la configuración utilizada por el recurso global activo, de tipo string.					

Operación **IApplication::GlobalResourcesFile**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**string**			
documentación	Establece el nombre de archivo/la URL del archivo XML de recursos globales (p. ej. GlobalResources.xml). Propiedades: El nombre del archivo XML, de tipo string. Nota: use la URL absoluta para el nombre de archivo de entrada. Una URL absoluta especifica la ubicación exacta del archivo p. ej. http://www.miSitioWeb.com/archivosxml/miEntradaxml.xml					

Operación **IApplication::ReportOptionalWarnings**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**bool**			
documentación	Habilita o deshabilita la notificación de advertencias. Propiedades: Devuelve un valor booleano. 'true' habilita la notificación de advertencias, 'false' la deshabilita.					

Operación **IApplication::ServerName**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**string**			

documentación	Establece el nombre de servidor para el servidor HTTP (este método falla en la edición RaptorXML Development Edition). Propiedades: El nombre del servidor, de tipo string. Genera una excepción RaptorXMLException cuando se produce un error.

Operación **IApplication::ServerPath**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**string**			
documentación	Establece la ruta de acceso en forma de URL para el servidor HTTP (este método falla en la edición RaptorXML Development Edition). Propiedades: La ruta de acceso del servidor, de tipo string. Genera una excepción RaptorXMLException cuando se produce un error.					

Operación **IApplication::ServerPort**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**ushort**			
documentación	Establece el puerto para el servidor HTTP (este método falla en la edición RaptorXML Development Edition). Propiedades: El puerto del servidor, de tipo ushort. Genera una excepción RaptorXMLException cuando se produce un error.					

Operación **IApplication::UserCatalog**

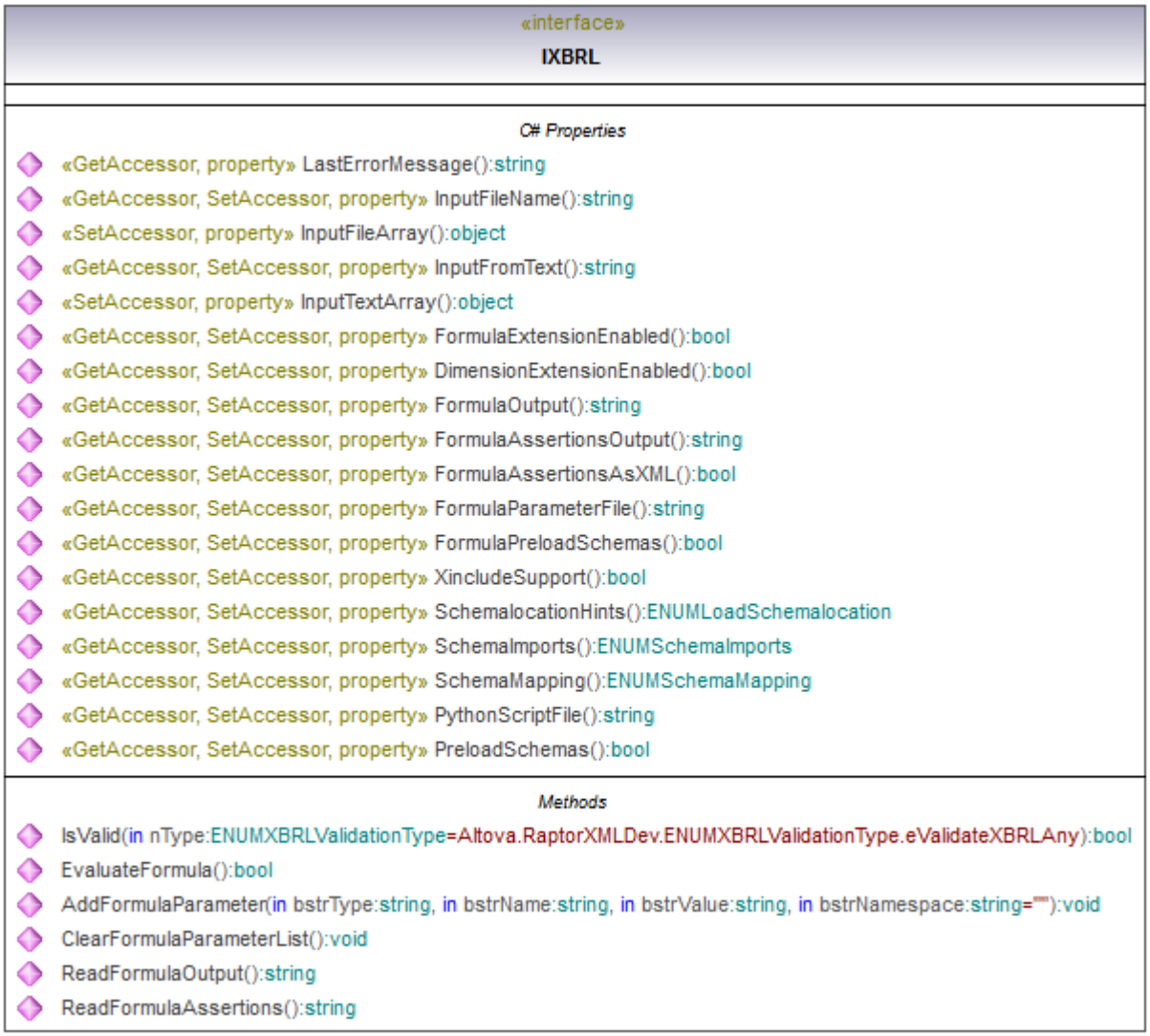
parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**string**			
documentación	Establece el nombre (en forma de URL) para el catálogo definido por el usuario (p. ej. CatalogoPersonal.xml). Propiedades: El nombre del catálogo definido por el usuario, de tipo string. Nota: use la URL absoluta para el nombre de archivo de entrada. Una URL absoluta especifica la ubicación exacta del archivo..					

© 2013 Altova GmbH

7.15 RaptorXMLDev_COM - IXBRL

Interfaz **IXBRL**

diagrama



documentación

Valida el documento XML o XBRL (*XBRL solo es compatible con RaptorXML+XBRL*) suministrado con DTD o esquemas XML externos o internos.

Comprueba si los documentos XML, DTD, XML Schema, XSLT y XQuery tienen un formato correcto.

Valida taxonomías XBRL y documentos XBRL con taxonomías XBRL (solo en RaptorXML+XBRL).

Operación **IXBRL::AddFormulaParameter**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	bstrType	**in**	**string**			
	bstrName	**in**	**string**			
	bstrValue	**in**	**string**			
	bstrNamespace	**in**	**string**			**""**
	return	**return**	**void**			
documentación	Añade un parámetro utilizado en el proceso de evaluación de fórmulas. Propiedades: 'bstrtype' es el tipo de datos del parámetro, de tipo string. 'bstrname' es el nombre del parámetro, de tipo string. 'bstrvalue' es el valor del parámetro, de tipo string. 'bstrnamespace' es el espacio de nombres del parámetro, de tipo string.					

Operación **IXBRL::ClearFormulaParameterList**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**void**			
documentación	Borra la lista de parámetros de fórmulas creada con el método AddFormulaParameter. Propiedades: ninguno.					

Operación **IXBRL::DimensionExtensionEnabled**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**bool**			
documentación	Habilita la validación con XBRL Dimension. Propiedades: Devuelve un valor booleano. 'true' habilita la validación con la extensión XBRL Dimension, 'false' la deshabilita.					

Operación **IXBRL::EvaluateFormula**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**bool**			
documentación	Evalúa las fórmulas XBRL de un archivo de instancia XBRL. Devuelve: 'true' si la operación finaliza correctamente, 'false' si falla. Si se produce un error durante la ejecución, use la operación getLastErrorMessage para obtener información adicional. Genera una excepción RaptorXMLException cuando se produce un error.					

Operación **IXBRL::FormulaAssertionsAsXML**

 © 2013 Altova GmbH

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**bool**			
documentación	Habilita el formato XML del archivo de aserciones cuando RaptorXML se ejecuta con las aserciones habilitadas. Propiedades: Devuelve un valor booleano. 'true' habilita el formato XML de salida, 'false' genera el resultado en formato JSON.					

Operación **IXBRL::FormulaAssertionsOutput**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**string**			
documentación	Establece la ubicación del archivo de salida de aserciones. Propiedades: La ruta de acceso completa del archivo de salida, de tipo string.					

Operación **IXBRL::FormulaExtensionEnabled**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**bool**			
documentación	Habilita la validación de extensiones de XBRL Formula. Propiedades: Devuelve un valor booleano. 'true' habilita la validación de extensiones de fórmulas, 'false' la deshabilita.					

Operación **IXBRL::FormulaOutput**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**string**			
documentación	Establece la ubicación donde debe guardarse el archivo de evaluación de fórmulas XBRL. Propiedades: La ruta de acceso completa del archivo de salida, de tipo string.					

Operación **IXBRL::FormulaParameterFile**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**string**			
documentación	Establece la ubicación donde debe guardarse el archivo de parámetros de fórmulas. Propiedades: La ruta de acceso completa del archivo de parámetros de fórmulas, de tipo string.					

Operación **IXBRL::FormulaPreloadSchemas**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**bool**			
documentación	Define la carga previa de los esquemas XBRL 2.1. Propiedades: 'true' carga de forma previa los esquemas XBRL. El valor predeterminado es 'false'.					

Operación **IXBRL::InputFileArray**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**object**			
documentación	Establece la matriz de archivos XBRL que se usarán como instancias/datos de entrada. Propiedades: Objeto que contiene las cadenas de las URL absolutas de cada archivo de entrada.					

Operación **IXBRL::InputFileName**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**string**			
documentación	Establece el nombre de archivo para el archivo de instancia XBRL de entrada. Propiedades: Una cadena que contiene la URL absoluta (o la ubicación base) del archivo de entrada.					

Operación **IXBRL::InputFromText**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**string**			
documentación	Suministra el contenido del documento XBRL de entrada en forma de texto. Propiedades: Contiene los datos XBRL como texto, de tipo string.					

Operación **IXBRL::InputTextArray**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**object**			
documentación	Establece la matriz de archivos que se usará como datos de entrada. Propiedades: objeto que contiene las cadenas de las URL absolutas de cada archivo de instancia XBRL de entrada.					

 © 2013 Altova GmbH

Operación **IXBRL::IsValid**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	nType	**in**	**ENUMXBRLValidationType**			**Altova.RaptorXMLDev.ENUMXBRLValidationType.eValidateXBRLAny**
	return	**return**	**bool**			
documentación	Resultado de la validación XBRL. Propiedades: 'nType' el valor de ENUMXBRLValidationType. Devuelve: 'true' si la operación se realiza correctamente, 'false' si falla. Genera una excepción RaptorXMLException cuando se produce un error. Si se produce un error durante la ejecución, use la operación LastErrorMessage para obtener información adicional.					

Operación **IXBRL::LastErrorMessage**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**string**			
documentación	Recupera el último mensaje de error del motor XBRL. Propiedades: Devuelve el texto del último mensaje de error, de tipo string.					

Operación **IXBRL::PreloadSchemas**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**bool**			
documentación	Define la carga previa de los esquemas XBRL 2.1. Valores: 'true' carga previamente los esquemas XBRL. El valor predeterminado es 'false'.					

Operación **IXBRL::PythonScriptFile**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**string**			

documentación	Establece el nombre de archivo (en forma de URL) del archivo de script Python. Parámetros: una URL absoluta que da la ubicación base del archivo, de tipo string.

Operación **IXBRL::ReadFormulaAssertions**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**string**			
documentación	Lee la evaluación de aserciones desde el archivo especificado. Parámetros: ninguno.					

Operación **IXBRL::ReadFormulaOutput**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**string**			
documentación	Lee el resultado de la evaluación de aserciones desde el archivo especificado. Parámetros: ninguno.					

Operación **IXBRL::SchemaImports**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**ENUMSchemaImports**			
documentación	Establece el formato de los errores de RaptorXML (Text, ShortXML, LongXML), depende de los literales de ENUMErrorFormat. Devuelve el valor del literal de enumeración seleccionado, de tipo ENUMErrorFormat.					

Operación **IXBRL::SchemalocationHints**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**ENUMLoadSchemalocation**			
documentación	Establece la ubicación desde la que se cargará el esquema, depende de ENUMLoadSchemaLocation. Devuelve el valor del literal de enumeración seleccionado, de tipo ENUMLoadSchemaLocation.					

Operación **IXBRL::SchemaMapping**

© 2013 Altova GmbH

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**ENUMSchemaMapping**			
documentación	Establece la ubicación desde la que se cargará el esquema, depende de ENUMSchemaMapping. Devuelve el valor del literal de enumeración seleccionado, de tipo ENUMSchemaMapping.					

Operación **IXBRL::XincludeSupport**

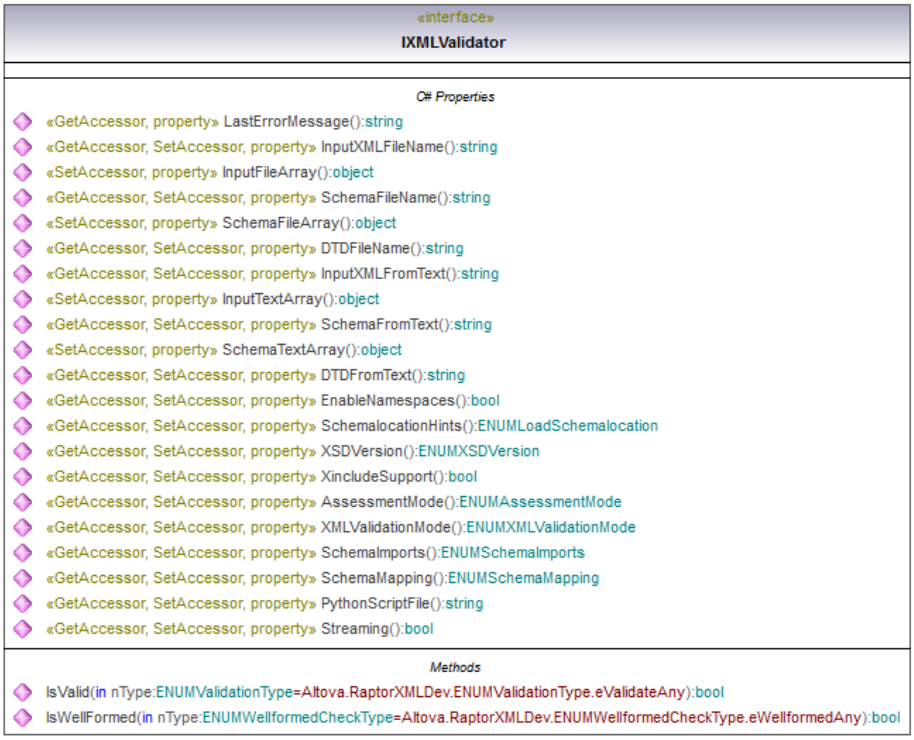
parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**bool**			
documentación	Habilita o deshabilita el uso de inclusiones XML (elementos XInclude). Parámetros: 'true' habilita los elementos XInclude include, 'false' los ignora.					

© 2013 Altova GmbH

7.16 RaptorXMLDev_COM - IXMLValidator

Interfaz **IXMLValidator**

diagrama



documentación

Descripción:
La interfaz IXMLValidator ofrece métodos para probar:

* Si un documento XML tiene un formato correcto.
* La validez de un documento XML usando una DTD o un esquema XML a los que se hace referencia dentro del documento XML.
* La validez de un documento XML usando una DTD o un esquema XML externos suministrados en el código.

Todos estos métodos devuelven los valores booleanos TRUE o FALSE.

Nota: si quiere que las cadenas de entrada se interpreten como direcciones URL, utilice rutas de acceso absolutas. Si utiliza rutas de acceso relativas, debe definir un mecanismo para resolver la ruta relativa en el módulo de llamada.

Operación **IXMLValidator::AssessmentMode**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**ENUMAssessmentMode**			

documentación

Establece el modo de evaluación del validador XML (Strict/Lax/Skip), depende de los literales de ENUMAssessmentMode.

Propiedades:
El valor del literal de enumeración seleccionado.

 © 2013 Altova GmbH

Operación **IXMLValidator::DTDFileName**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**string**			
documentación	Establece qué documento DTD externo se usa para la validación. Propiedades: Una URL absoluta que da la ubicación base de la DTD que debe usarse, de tipo string. Una URL absoluta especifica la ubicación exacta del archivo p. ej. http://www.miSitioWeb.com/archivosxml/miEntradaxml.xml.					

Operación **IXMLValidator::DTDFromText**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**string**			
documentación	Establece el valor de texto para la DTD externa. Propiedades: Contiene la DTD en forma de texto, de tipo string.					

Operación **IXMLValidator::EnableNamespaces**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**bool**			
documentación	Habilita el procesamiento preparado para espacios de nombres, lo cual es práctico para comprobar si hay errores en la instancia XML debido al uso incorrecto de espacios de nombres. Propiedades: Devuelve un valor booleano. 'true' habilita el procesamiento preparado para espacios de nombres, 'false' lo deshabilita.					

Operación **IXMLValidator::InputFileArray**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**object**			
documentación	Establece la matriz de archivos XML que se usarán como datos de entrada. Propiedades: Objeto que contiene las cadenas de las URL absolutas de cada archivo XML.					

Operación **IXMLValidator::InputTextArray**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**object**			

documentación	Establece la matriz de archivos de texto que se usarán como datos de entrada. Propiedades: Objeto que contiene las cadenas de las URL absolutas de cada archivo de texto de entrada.

Operación **IXMLValidator::InputXMLFileName**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**string**			
documentación	Establece el nombre de archivo (como URL) del archivo XML de entrada. Propiedades: Una URL absoluta que da la ubicación base del archivo XML, de tipo string. Una URL absoluta especifica la ubicación exacta del archivo p. ej. http://www.miSitioWeb.com/archivosxml/miEntradaxml.xml.					

Operación **IXMLValidator::InputXMLFromText**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**string**			
documentación	Suministra el contenido del documento XML de entrada en forma de texto. Propiedades: Contiene los datos XML, de tipo string. Nota: use la URL absoluta para el nombre de archivo de entrada. Una URL absoluta especifica la ubicación exacta del archivo p. ej. http://www.miSitioWeb.com/archivosxml/miEntradaxml.xml					

Operación **IXMLValidator::IsValid**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	nType	**in**	**ENUMValidationType**			**Altova.RaptorXMLDev.ENUMValidationType.eValidateAny**
	return	**return**	**bool**			

© 2013 Altova GmbH

documentación	Resultado de la validación XML. Propiedades: 'nType' el valor de ENUMValidationType. Devuelve: 'true' si la operación finaliza correctamente, 'false' si falla. Genera una excepción RaptorXMLException cuando se produce un error. Si se produce un error durante la ejecución, use la operación LastErrorMessage para obtener información adicional.

Operación **IXMLValidator::IsWellFormed**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	nType	**in**	**ENUMWellformedCheckType**			**Altova.RaptorXMLDev.ENUMWellformedCheckType.eWellformedAny**
	return	**return**	**bool**			
documentación	Resultado de la comprobación de formato especificada por el valor de ENUMWellformedCheckType. Propiedades: 'nType' es el valor de ENUMWellformedCheckType. Devuelve: 'true' si la operación finaliza correctamente, 'false' si falla. Genera una excepción RaptorXMLException cuando se produce un error. Si se produce un error durate la ejecución, use LastErrorMessage para obtener información adicional.					

Operación **IXMLValidator::LastErrorMessage**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**string**			
documentación	Recupera el último mensaje de error del motor XML. Propiedades: Devuelve el texto del último mensaje de error, de tipo string.					

Operación **IXMLValidator::PythonScriptFile**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**string**			

documentación	Establece el nombre de archivo (como URL) del archivo de script Python. Devuelve una URL absoluta que da la ubicación base del archivo, de tipo string.

Operación **IXMLValidator::SchemaFileArray**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**object**			
documentación	Establece la matriz de archivos que se usarán como esquemas XML externos. Propiedades: La matriz de URL absolutas que contiene la ubicación de los esquemas, de tipo object.					

Operación **IXMLValidator::SchemaFileName**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**string**			
documentación	Establece el nombre de archivo de los esquemas XML externos. Propiedades: La URL absoluta de la ubicación base del esquema, de tipo string.					

Operación **IXMLValidator::SchemaFromText**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**string**			
documentación	Suministra el contenido del esquema XML externo en forma de texto. Propiedades: La cadena que contiene el esquema XML como texto, de tipo string.					

Operación **IXMLValidator::SchemaImports**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**ENUMSchemaImports**			
documentación	Especifica el comportamiento de los elementos xs:import y cada uno tiene un atributo de espacio de nombres opcional y un atributo schemaLocation opcional. Devuelve el método de importación del esquema de tipo EnumSchemaImports					

Operación **IXMLValidator::SchemalocationHints**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**ENUMLoadSchemalocation**			

 © 2013 Altova GmbH

documentación	Establece la ubicación desde la que se cargará el esquema, depende de ENUMLoadSchemaLocation. Devuelve el valor del literal de enumeración seleccionado, de tipo ENUMLoadSchemaLocation.

Operación **IXMLValidator::SchemaMapping**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**ENUMSchemaMapping**			
documentación	Establece la ubicación desde la que se cargará el esquema, depende de ENUMSchemaMapping. Devuelve el valor del literal de enumeración seleccionado, de tipo ENUMSchemaMapping.					

Operación **IXMLValidator::SchemaTextArray**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**object**			
documentación	Establece la colección de cadenas que usarán como esquemas XML externos. Propiedades: Una matriz de URL absolutas que contienen la ubicación de los esquemas, de tipo object.					

Operación **IXMLValidator::Streaming**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**bool**			
documentación	Habilita la validación por transmisión de secuencias. En el modo de transmisión por secuencias, los datos almacenados en memoria se minimizan y se procesan más rápido. Valores: 'true' habilita la validación por transmisión de secuencias, 'false' la deshabilita. El valor predeterminado es 'true'.					

Operación **IXMLValidator::XincludeSupport**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**bool**			
documentación	Habilita o deshabilita el uso de inclusiones XML (elementos XInclude). Propiedades: Devuelve un valor booleano. 'true' habilita los elementos XInclude include, 'false' los ignora.					

Operación **IXMLValidator::XMLValidationMode**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**ENUMXMLValidationMode**			
documentación	Establece el modo de validación XML, depende de ENUMXMLValidationMode. Devuelve el valor del literal de enumeración seleccionado, de tipo ENUMXMLValidationMode.					

Operación **IXMLValidator::XSDVersion**

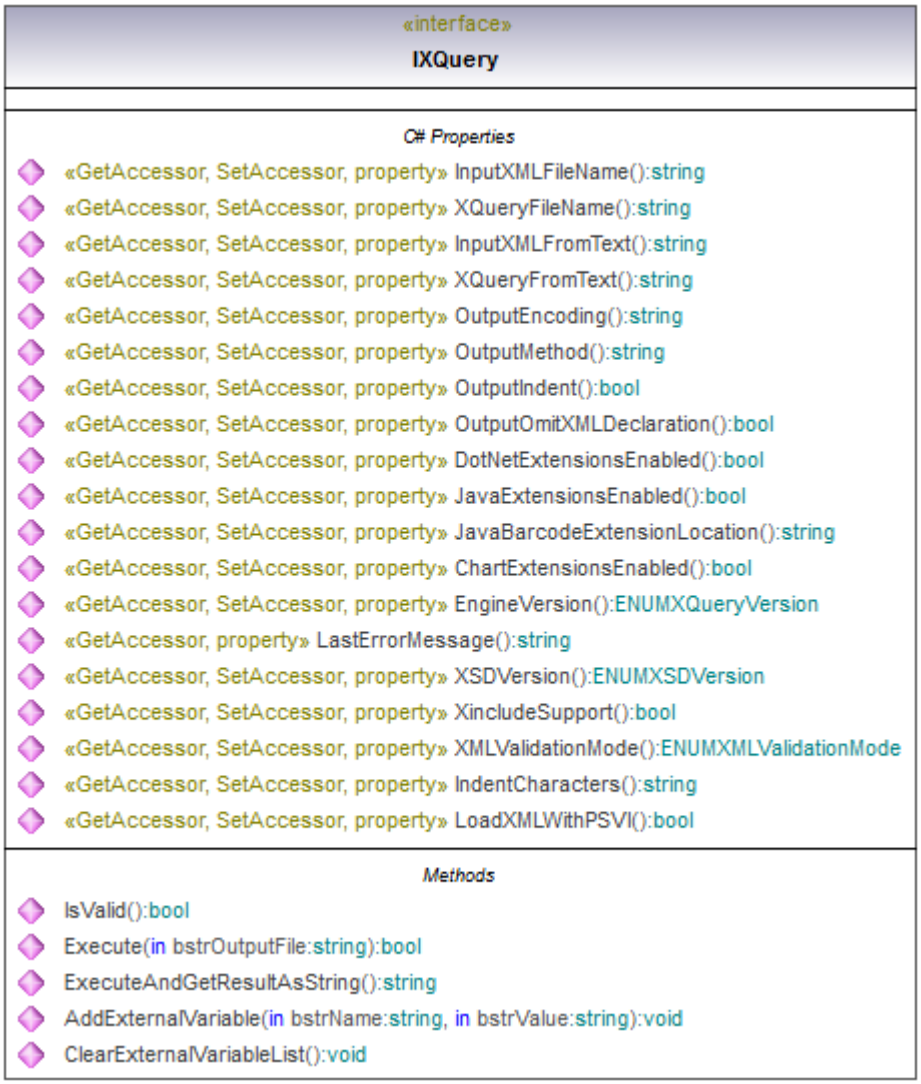
parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**ENUMXSDVersion**			
documentación	Define la versión de XML Schema que se usará para validar el documento. Propiedades: La versión de XML Schema definida por el literal de EnumXSDVersion, de tipo EnumXSDVersion.					

Documentación UML generada con el editor UML **UModel**. **Más información en http://www.altova.com/es/umodel** 06/07/13 15:48:10

7.17 RaptorXMLDev_COM - IXQuery

Interfaz **IXQuery**

diagrama



documentación

La interfaz IXQuery ofrece métodos y propiedades para ejecutar una transformación XQuery 1.0/3.0 usando el motor RaptorXML.

Los resultados se pueden guardar en un archivo o se pueden devolver en forma de cadena. El objeto también permite pasar variables XQuery externas al documento XQuery. Las URL de los archivos XQuery y XML se pueden suministrar como cadenas de texto a través de las propiedades del objeto.

Otra opción es construir los documentos XML y XQuery dentro del código como cadenas de texto.

Nota: si quiere que las cadenas de entrada se interpreten como direcciones URL, utilice rutas de acceso absolutas. Si utiliza rutas de acceso relativas, debe definir un mecanismo para resolver la ruta relativa en el módulo de llamada.

Operación **IXQuery::AddExternalVariable**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	bstrName	**in**	**string**			
	bstrValue	**in**	**string**			
	return	**return**	**void**			
documentación	Añade el nombre y el valor de una variable externa nueva. Propiedades: 'bstrName' es el nombre de variable y es un QName válido, de tipo string. 'bstrValue' es el valor de la variable, de tipo string. Cada variable externa y su valor deben especificarse en una llamada al método distinta. Las variables deben declararse en el documento XQuery y, si quiere, con una declaración de tipo. Independientemente de la declaración de tipo que tenga la variable externa en el documento XQuery, el valor de variable enviado a la operación AddExternalVariable no necesita llevar delimitadores especiales.					

Operación **IXQuery::ChartExtensionsEnabled**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**bool**			
documentación	Habilita o deshabilita las extensiones de Altova para gráficos. Propiedades: Devuelve un valor booleano. 'true' habilita las extensiones para gráficos, 'false' las deshabilita.					

Operación **IXQuery::ClearExternalVariableList**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**void**			
documentación	Borra la lista de variables externas creada con el método AddExternalVariable.					

Operación **IXQuery::DotNetExtensionsEnabled**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**bool**			
documentación	Habilita o deshabilita las extensiones de Visual Studio .NET. Propiedades: Devuelve un valor booleano 'true' habilita las extensiones .NET, 'false' las deshabilita.					

Operación **IXQuery::EngineVersion**

 © 2013 Altova GmbH

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**ENUMXQueryVersion**			
documentación	Establece la versión XQuery que se utilizará: XQuery 1.0/3.0. Propiedades: El número de versión establecido por el literal de la enumeración EnumQueryVersion: eVersion10 o eVersion30.					

Operación **IXQuery::Execute**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	bstrOutputFile	**in**	**string**			
	return	**return**	**bool**			
documentación	Ejecuta la transformación XQuery (dependiendo del valor de ENUMQueryVersion) y guarda el resultado en un archivo de salida. Propiedades: 'bstrOutputFile' es la ruta de acceso (y el nombre de archivo) del archivo de salida. Si se produce un error durante la ejecución, use la operación LastErrorMessage para obtener información adicional. Genera una excepción RaptorXMLException cuando se produce un error.					

Operación **IXQuery::ExecuteAndGetResultAsString**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**string**			
documentación	Ejecuta el XQuery y devuelve el resultado en forma de cadena de texto. Propiedades: Devuelve el resultado de la transformación en forma de cadena de texto. Si se produce un error durante la ejecución, use la operación getLastErrorMessage para obtener información adicional. Genera una excepción RaptorXMLException cuando se produce un error.					

Operación **IXQuery::IndentCharacters**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**string**			
documentación	Establece la cadena de caracteres que se usará como sangría. Devuelve el carácter de sangría, de tipo string.					

Operación **IXQuery::InputXMLFileName**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**string**			
documentación	Establece el nombre de archivo (en forma de URL) de la instancia XML de entrada que se debe transformar. Propiedades: El nombre / la URI del archivo XML, de tipo string. Nota: use la URL absoluta para el nombre de archivo de entrada. Una URL absoluta especifica la ubicación exacta del archivo p. ej. http://www.miSitioWeb.com/archivosxml/miEntradaxml.xml					

Operación **IXQuery::InputXMLFromText**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**string**			
documentación	Suministra el contenido del documento XML de entrada en forma de texto. Propiedades: Los datos XML de entrada, de tipo string. Nota: use la URL absoluta para el nombre de archivo de entrada. Una URL absoluta especifica la ubicación exacta del archivo p. ej. http://www.miSitioWeb.com/archivosxml/miEntradaxml.xml					

Operación **IXQuery::IsValid**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**bool**			
documentación	Resultado de la validación XQuery 1.0/3.0 (dependiendo del valor de ENUMQueryVersion). Propiedades: Devuelve un valor booleano. 'true' si la operación finaliza correctamente, 'false' si falla. Genera una excepción RaptorXMLException cuando se produce un error.					

Operación **IXQuery::JavaBarcodeExtensionLocation**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**string**			
documentación	Habilita o deshabilita las extensiones de códigos de barras Java. Propiedades: Devuelve un valor booleano. 'true' habilita las extensiones de códigos de barras Java, 'false' las deshabilita.					

© 2013 Altova GmbH

Operación **IXQuery::JavaExtensionsEnabled**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**bool**			
documentación	Habilita o deshabilita las extensiones Java. Propiedades: Devuelve un valor booleano. 'true' habilita las extensiones Java, 'false' las deshabilita.					

Operación **IXQuery::LastErrorMessage**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**string**			
documentación	Recupera el último mensaje de error del motor XQuery. Propiedades: Devuelve el texto del último mensaje de error, de tipo string.					

Operación **IXQuery::LoadXMLWithPSVI**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**bool**			
documentación	Habilita o deshabilita la opción "Cargar conjunto de información posterior a la validación con esquema". Devuelve el valor booleano. Valores: 'true' habilita el conjunto de información posterior a la validación con esquema, 'false' lo deshabilita.					

Operación **IXQuery::OutputEncoding**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**string**			
documentación	Establece la codificación del documento de resultados. Propiedades: El nombre de codificación (p. ej. : UTF-8, UTF-16, ASCII, 8859-1, 1252), de tipo string.					

Operación **IXQuery::OutputIndent**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**bool**			

documentación	Habilita o deshabilita la opción de sangría para el documento de resultados. Propiedades: Devuelve un valor booleano. 'true' habilita la opción de sangría, 'false' la deshabilita.

Operación **IXQuery::OutputMethod**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**string**			
documentación	Establece el método de serialización para el documento de resultados. Propiedades: El método de serialización, de tipo string. Valores: xml, xhtml, html, text.					

Operación **IXQuery::OutputOmitXMLDeclaration**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**bool**			
documentación	Habilita o deshabilita la generación de la declaración XML en el documento de resultados. Propiedades: Devuelve un valor booleano. 'true' omite la declaración XML, 'false' la incluye.					

Operación **IXQuery::XincludeSupport**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**bool**			
documentación	Habilita o deshabilita el uso de inclusiones XML (elementos XInclude). Devuelve un valor booleano. Valores: 'true' habilita los elementos XInclude include, 'false' los ignora.					

Operación **IXQuery::XMLValidationMode**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**ENUMXMLValidationMode**			

© 2013 Altova GmbH

documentación	Establece el modo de validación XML, depende de ENUMXMLValidationMode. Devuelve el valor del literal de enumeración seleccionado, de tipo ENUMXMLValidationMode.

Operación **IXQuery::XQueryFileName**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**string**			
documentación	Establece el nombre de archivo del documento XQuery. Propiedades: La URL absoluta que da la ubicación base del archivo XQuery, de tipo string. Una URL absoluta especifica la ubicación exacta del archivo p. ej. http://www.miSitioWeb.com/archivosxml/miEntradaxml.xml					

Operación **IXQuery::XQueryFromText**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**string**			
documentación	Suministra el contenido de la instrucción XQuery en forma de texto. Propiedades: La instrucción XQuery, de tipo string.					

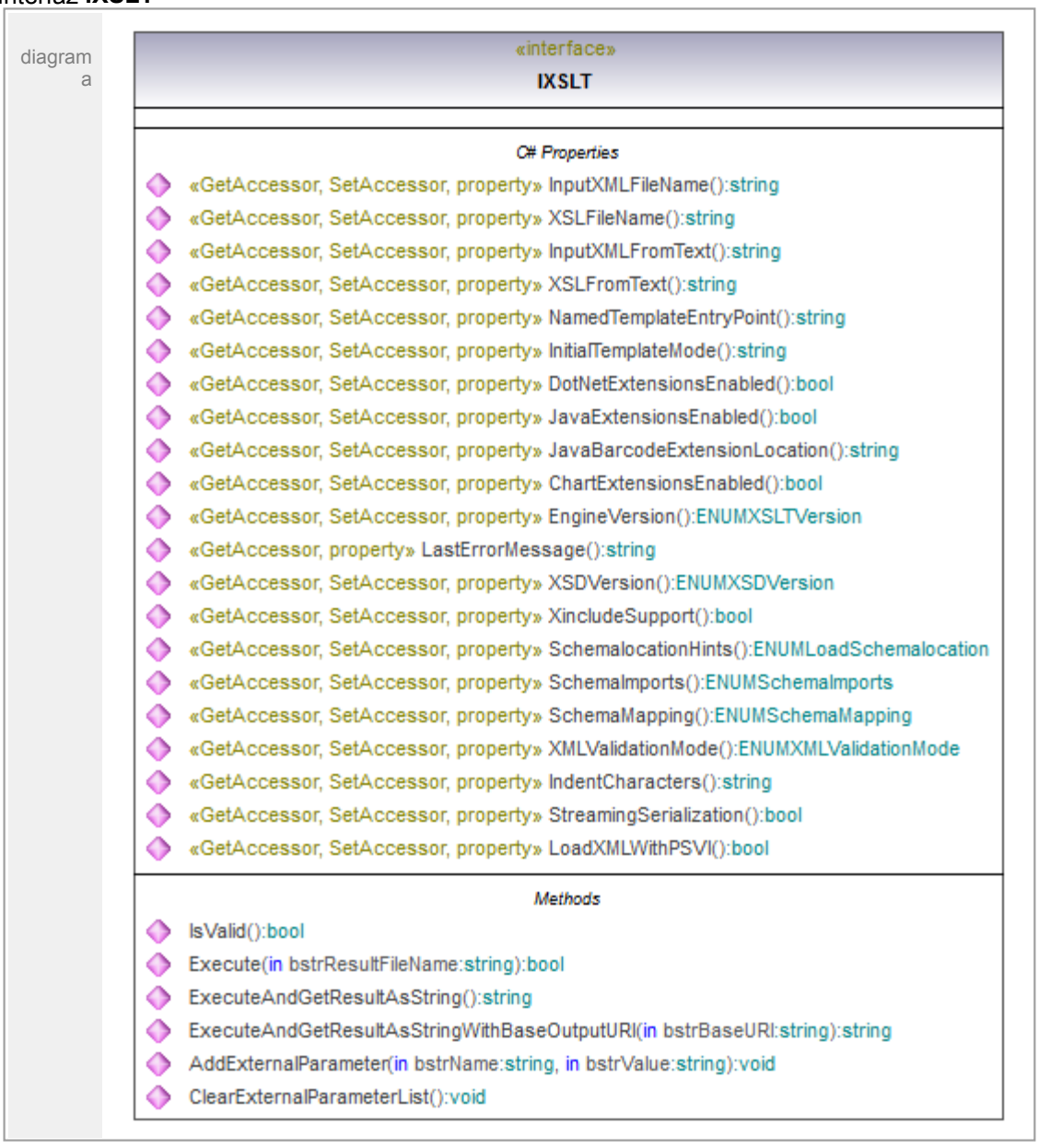
Operación **IXQuery::XSDVersion**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**ENUMXSDVersion**			
documentación	Define la versión de XML Schema que se usará para validar el documento. Devuelve la versión de XML Schema establecida por el literal de EnumXSDVersion, de tipo EnumXSDVersion.					

© 2013 Altova GmbH

7.18 RaptorXMLDev_COM - IXSLT

Interfaz **IXSLT**



 © 2013 Altova GmbH

documentación	El objeto IXSLT ofrece métodos y propiedades para ejecutar una transformación XSLT 1.0/2.0/3.0 con el procesador RaptorXML. Los resultados se pueden guardar en un archivo o se pueden devolver en forma de cadena. El objeto también permite pasar parámetros XSLT a la hoja de estilos XSLT. Las URL de los archivos XSLT y XML se pueden suministrar como cadenas de texto a través de las propiedades del objeto. Otra opción es construir los documentos XML y XSLT dentro del código como cadenas de texto. Nota: si quiere que las cadenas de entrada se interpreten como direcciones URL, utilice rutas de acceso absolutas. Si utiliza rutas de acceso relativas, debe definir un mecanismo para resolver la ruta relativa en el módulo de llamada. El motor RaptorXML se puede usar en modo de compatibilidad con versiones anteriores para poder procesar hojas de estilos XSLT 1.0. El resultado, sin embargo, puede ser algo diferente al que se conseguiría con el motor XSLT 1.0 y la misma hoja de estilos.

Operación **IXSLT::AddExternalParameter**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	bstrName	**in**	**string**			
	bstrValue	**in**	**string**			
	return	**return**	**void**			
documentación	Añade el nombre y el valor de un parámetro externo nuevo. Propiedades: 'bstrName' es el nombre del parámetro y es un QName válido, de tipo string. 'bstrValue' es el valor del parámetro, de tipo string. Cada parámetro externo y su valor debe especificarse en una llamada al método distinta. Los parámetros deben declararse en el documento XQuery y, si quiere, con una declaración de tipo. Independientemente de la declaración de tipo que tenga el parámetro externo en el documento XSLT, el valor de parámetro enviado a la operación AddExternalParameter no necesita llevar delimitadores especiales.					

Operación **IXSLT::ChartExtensionsEnabled**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**bool**			
documentación	Habilita o deshabilita las extensiones de Altova para gráficos. Propiedades: Devuelve un valor booleano. 'true' habilita las extensiones para gráficos, 'false' las deshabilita					

Operación **IXSLT::ClearExternalParameterList**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**void**			
documentación	Borra la lista de parámetros externos creada con el método AddExternalParameter .					

Operación **IXSLT::DotNetExtensionsEnabled**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**bool**			
documentación	Habilita o deshabilita las extensiones Visual Studio .NET. Propiedades: Devuelve un valor booleano 'true' habilita las extensiones .NET, 'false' las deshabilita.					

Operación **IXSLT::EngineVersion**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**ENUMXSLTVersion**			
documentación	Establece la versión XSLT que se utilizará (XSLT1.0/2.0/3.0). Propiedades: El número de versión establecido por el literal de la enumeración EnumXSLTVersion: eVersion10, eVersion20 o eVersion30.					

Operación **IXSLT::Execute**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	bstrResultFilinName		**string**			
	return	**return**	**bool**			
documentación	Ejecuta la transformación XSLT (dependiendo del valor de ENUMXSLTVersion) y guarda el resultado en un archivo de salida. Propiedades: 'bstrResultFile' es la ruta de acceso (y el nombre de archivo) del archivo de salida. Si se produce un error durante la ejecución, use la operación LastErrorMessage para obtener información adicional. Genera una excepción RaptorXMLException cuando se produce un error.					

Operación **IXSLT::ExecuteAndGetResultAsString**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**string**			

© 2013 Altova GmbH

documentación	Ejecuta el XSLT y devuelve el resultado en forma de cadena. Propiedades: Devuelve el resultado de la transformación en forma de cadena de texto. Si se produce un error durante la ejecución, use la operación LastErrorMessage para obtener información adicional. Genera una excepción RaptorXMLException cuando se produce un error.

Operación **IXSLT::ExecuteAndGetResultAsStringWithBaseOutputURI**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	bstrBaseURI	**in**	**string**			
	return	**return**	**string**			
documentación	Ejecuta el XSLT y devuelve el resultado en forma de cadena en la ubicación definida por el URI base. Parámetros: ninguno. Si se produce un error durante la ejecución, use la operación getLastErrorMessage para obtener información adicional. Genera una excepción RaptorXMLException cuando se produce un error.					

Operación **IXSLT::IndentCharacters**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**string**			
documentación	Establece la cadena de caracteres que se usará como sangría. Devuelve el carácter de sangría, de tipo string.					

Operación **IXSLT::InitialTemplateMode**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**string**			
documentación	Establece el modo inicial para el procesamiento XSLT. Propiedades: El nombre del modo inicial que desea usar, de tipo string. Se procesarán las plantillas que tengan este modo. Por ejemplo SetInitialTemplateMode="MiModo". Nota: la transformación siempre debe producirse después de asignar los documentos XML y XSLT.					

Operación **IXSLT::InputXMLFileName**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**string**			
documentación	Establece el nombre de archivo (en forma de URL) de la instancia XML de entrada que se debe transformar. Propiedades: El nombre / la URL del archivo XML, de tipo string. Nota: use la URL absoluta para el nombre de archivo de entrada. Una URL absoluta especifica la ubicación exacta del archivo p. ej. http://www.miSitioWeb.com/archivosxml/miEntradaxml.xml					

Operación **IXSLT::InputXMLFromText**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**string**			
documentación	Suministra el contenido del documento XML de entrada en forma de texto. Propiedades: Los datos XML de entrada, de tipo string. Nota: use la URL absoluta para el nombre de archivo de entrada. Una URL absoluta especifica la ubicación exacta del archivo p. ej. http://www.miSitioWeb.com/archivosxml/miEntradaxml.xml					

Operación **IXSLT::IsValid**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**bool**			
documentación	Resultado de la validación XSLT (depende del valor de ENUMXSLTVersion) Propiedades: Devuelve un valor booleano. 'true' si la operación finaliza correctamente, 'false' si falla. Genera una excepción RaptorXMLException cuando se produce un error.					

Operación **IXSLT::JavaBarcodeExtensionLocation**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**string**			
documentación	Define la ubicación del archivo de la extensión de códigos de barras para Java. Propiedades: Ubicación del archivo de extensión, de tipo string. Nota: use la URL absoluta para el nombre de archivo de entrada. Una URL absoluta especifica la ubicación exacta del archivo. p. ej. http://www.myWebsite.com/xmlfiles/miEntradaxml.xml					

© 2013 Altova GmbH

Operación **IXSLT::JavaExtensionsEnabled**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**bool**			
documentación	Habilita o deshabilita las extensiones Java. Parámetros: Devuelve un valor booleano. Valores: 'true' habilita las extensiones Java, 'false' las deshabilita.					

Operación **IXSLT::LastErrorMessage**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**string**			
documentación	Recupera el último mensaje de error del motor XSLT. Propiedades: Devuelve el texto del último mensaje de error, de tipo string.					

Operación **IXSLT::LoadXMLWithPSVI**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**bool**			
documentación	Habilita o deshabilita la opción "Cargar conjunto de información posterior a la validación con esquema". Devuelve el valor booleano. Valores: 'true' habilita el conjunto de información posterior a la validación con esquema, 'false' lo deshabilita.					

Operación **IXSLT::NamedTemplateEntryPoint**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**string**			
documentación	Establece el punto de entrada de la plantilla con nombre. Propiedades: El nombre del nodo a partir del cual debe empezar el procesamiento, de tipo string.					

Operación **IXSLT::SchemaImports**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**ENUMSchemaImports**			

documentación	Define el comportamiento de los elementos xs:import y cada uno tiene un atributo de espacio de nombres opcional y un atributo schemaLocation opcional, depende de ENUMSchemaImports.

Operación **IXSLT::SchemalocationHints**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**ENUMLoadSchemalocation**			
documentación	Establece la ubicación desde la que se cargará el esquema, depende de ENUMLoadSchemaLocation. Devuelve el valor del literal de enumeración seleccionado, de tipo ENUMLoadSchemaLocation.					

Operación **IXSLT::SchemaMapping**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**ENUMSchemaMapping**			
documentación	Establece la ubicación desde la que se cargará el esquema, depende de ENUMSchemaMapping. Devuelve el valor del literal de enumeración seleccionado, de tipo ENUMSchemaMapping.					

Operación **IXSLT::StreamingSerialization**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**bool**			
documentación	Habilita la serialización de secuencias de datos. En el modo de transmisión por secuencias, los datos almacenados en memoria se minimizan y se procesan más rápido. Devuelve un valor booleano: 'true' habilita la serialización por secuencias, 'false' la deshabilita. El valor predeterminado es 'true'.					

Operación **IXSLT::XincludeSupport**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**bool**			
documentación	Habilita o deshabilita el uso de inclusiones XML (elementos XInclude). Devuelve un valor booleano. 'true' habilita los elementos XInclude include, 'false' los ignora.					

Operación **IXSLT::XMLValidationMode**

 © 2013 Altova GmbH

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**ENUMXMLValidationMode**			
documentación	Establece el modo de validación XML, depende de ENUMXMLValidationMode. Devuelve el valor del literal de enumeración seleccionado, de tipo ENUMXMLValidationMode.					

Operación **IXSLT::XSDVersion**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**ENUMXSDVersion**			
documentación	Define la versión de XML Schema que se usará para validar el documento. Devuelve la versión de XML Schema establecida por el literal de EnumXSDVersion, de tipo EnumXSDVersion.					

Operación **IXSLT::XSLFileName**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**string**			
documentación	Establece la ruta de acceso / la URL necesaria para ubicar el archivo XSLT que debe usarse para la transformación. Propiedades: La ruta de acceso / el nombre de archivo del archivo XSLT, de tipo string. Nota: use la URL absoluta para el nombre de archivo de entrada. Una URL absoluta especifica la ubicación exacta del archivo p. ej. http://www.miSitioWeb.com/archivosxml/miEntradaxml.xml.					

Operación **IXSLT::XSLFromText**

parámetro	nombre	dirección	tipo	tipo modificador	multiplicidad	predeterminado
	return	**return**	**string**			
documentación	Establece el nombre de archivo XSLT que se debe usar para la transformación. Propiedades: El nombre del archivo de texto XML del archivo XSLT, de tipo string.					

© 2013 Altova GmbH

Altova RaptorXML 2013

Información sobre motores XSLT y XQuery

8 Información sobre motores XSLT y XQuery

Los motores XSLT y XQuery de RaptorXML siguen las especificaciones del W3C y, por tanto, son más estrictos que otros motores anteriores de Altova, como los del procesador descatalogado AltovaXML. Por consiguiente, RaptorXML 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
- XSLT 2.0
- XSLT 3.0
- XQuery 1.0
- XQuery 3.0
- Funciones XQuery 1.0 y XPath 2.0
- Funciones XQuery y XPath 3.0

© 2013 Altova GmbH

8.1 XSLT 1.0

El motor XSLT 1.0 de RaptorXML (disponible en todas las ediciones) 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 ` ` (la referencia de carácter decimal para un espacio de no separación) se inserta como ` ` en el código HTML.

8.2 XSLT 2.0

Temas de este apartado:

- Especificaciones con las que cumple el motor
- Compatibilidad con versiones antiguas
- Espacios de nombres
- Compatibilidad con esquemas
- Comportamiento propio de esta implementación

Especificaciones
El motor XSLT 2.0 de RaptorXML (disponible en todas las ediciones) 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 solamente es relevante cuando se utiliza el motor XSLT 2.0 (parámetro de la interfaz de la línea de comandos `--xslt=2`) para procesar una hoja de estilos 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="<%NS-FN%>"
    ...
</xsl:stylesheet>
```

Es necesario tener en cuenta que:

 © 2013 Altova GmbH

- 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')`.

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 del comportamiento 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 (XSLT 2.0, XPath 2.0 y funciones 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`.

`unparsed-text-available`
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`.

Nota: estos valores de codificación estaban implementados en el ya descatalogado AltovaXML pero ya no se utilizan (son obsoletos): `base16tobinary`, `base64tobinary`, `binarytobase16` y `binarytobase64`.

Funciones XPath 2.0
Consulte el apartado Funciones XQuery 1.0 y XPath 2.0.

© 2013 Altova GmbH

8.3 XSLT 3.0

El motor XSLT 3.0 de RaptorXML (disponible en todas las ediciones) cumple con el borrador de trabajo XSLT 2.0 del 10 de julio de 2012 y con la recomendación XPath 2.0 del 8 de enero de 2013, ambos del W3C.

El motor XSLT 2.0 tiene las mismas características de implementación que el motor XSLT 2.0. Pero además ofrece las nuevas funciones nuevas `xsl:evaluate`, `xsl:try` y `xsl:catch` y es compatible con funciones y operadores XPath y XQuery 3.0 y con la especificación XPath 3.0.

8.4 XQuery 1.0

Temas de este apartado:

- Especificaciones con las que cumple el motor
- Compatibilidad con esquemas
- Codificación
- Espacios de nombres
- Fuentes XML y validación
- Comprobación de tipos estática y dinámica
- Módulos biblioteca
- Módulos externos
- Intercalaciones
- Precisión de datos numéricos
- Compatibilidad con instrucciones XQuery
- Compatibilidad con funciones XQuery

Especificaciones compatibles
El motor XQuery 1.0 de RaptorXML (disponible en todas las ediciones) 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.

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 Scheme	`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`

© 2013 Altova GmbH

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 (ver tabla) 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";
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. Para usar una intercalación concreta, indique su URI tal y como aparece en la lista de intercalaciones compatibles. 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.

 © 2013 Altova GmbH

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.

Compatibilidad con funciones XQuery
Para más información sobre el comportamiento de las funciones XQuery 1.0, consulte el apartado Funciones XPath 2.0 y XQuery 1.0.

8.5 XQuery 3.0

El motor XQuery 3.0 de RaptorXML (disponible en todas las ediciones) cumple con la recomendación XQuery 3.0 del 8 de enero de 2013 del W3C y es compatible con Funciones XPath y XQuery 3.0

Tiene las mismas características de implementación que el motor XQuery 1.0.

© 2013 Altova GmbH

8.6 Funciones XQuery 1.0 y XPath 2.0

Temas de este apartado:

- Especificaciones
- Aspectos generales (incluidas las intercalaciones compatibles)
- `base-uri`
- `collection`
- `current-date, current-dateTime, current-time`
- `doc`
- `id`
- `in-scope-prefixes`
- `normalize-unicode`
- `resolve-uri`
- `static-base-uri`

Especificaciones
Las funciones XQuery 1.0 y XPath 2.0 compatibles con RaptorXML (en todas las ediciones) cumplen con la recomendación XQuery 1.0 and XPath 2.0 Functions and Operators del 14 de diciembre de 2010 del W3C.

Aspectos generales
A continuación encontrará una lista del comportamiento de ciertas funciones propias de esta implementación. Es importante tener en cuenta que:

- El espacio de nombres predeterminado de las funciones cumple con las normas del estándar. Por tanto, puede llamar a las funciones sin usar un prefijo.
- En general, si una función espera una secuencia de un elemento como argumento y se suministra una secuencia con más de un elemento, se devuelve un error.
- Todas las comparaciones de cadenas se hacen usando la intercalación de puntos de código Unicode.
- Los resultados que son QName se serializan de esta forma `[prefijo:]nombrelocal`.

Precisión de xs:decimal
El término *precisión* hace referencia al número de dígitos del número y la especificación exige un mínimo de 18 dígitos. Para las operaciones de división que dan un resultado de tipo `xs:decimal`, la precisión es de 19 dígitos después del punto decimal sin redondear.

Uso horario implícito
Cuando hay que comparar dos valores `date`, `time` o `dateTime`, es necesario conocer el uso horario de los valores que se comparan. Cuando el uso horario no se conoce de forma explícita, se usa el uso horario implícito, que se toma del reloj del sistema y su valor se puede comprobar con la función `fn:implicit-timezone()`.

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 aparecen más abajo. Para usar una intercalación determinada, aporte su URI tal y como aparece en la lista de intercalaciones compatibles. Las comparaciones de cadenas de texto (incluidas las necesarias para las funciones `fn:max` y `fn:min`) se harán según la intercalación especificada. Si no se especifica la opción de intercalación, se usa la intercalación de puntos de código Unicode predeterminada.

Idioma	Identificadores URI
`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 Bokmal	`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 se dejó de utilizar en XPath 2.0. Sin embargo, RaptorXML es compatible con el uso de ejes de espacio de nombres. Para acceder a la información sobre el espacio de nombres con mecanismos de XPath 2.0, utilice las funciones `fn:in-scope-prefixes()`, `fn:namespace-uri()` y `fn:namespace-uri-for-prefix()`.

`base-uri`

- Si se usan entidades externas en el documento XML de origen y si se especifica un nodo de la entidad externa como argumento de entrada de la función `base-uri()`, se utiliza el URI base del documento XML y no el URI base de la entidad externa.
- El URI base de un nodo del documento XML se puede modificar usando el atributo

 © 2013 Altova GmbH

`xml:base`.

---

collection

- El argumento es un URI relativo que se resuelve con el URI base actual.
- Si el URI resuelto identifica un archivo XML, este archivo XML se trata como un catálogo que hace referencia a una colección de archivos. Este archivo debe tener este formato:

```
<collection>
    <doc href="uri-1" />
    <doc href="uri-2" />
    <doc href="uri-3" />
 </collection>
```

  Se cargan los archivos a los que hacen referencia los atributos `href` y sus nodos de documento se devuelven en forma de secuencia.
- Si el URI resuelto no encuentra ningún archivo XML con la estructura de catálogo descrita, la cadena del argumento (que admite comodines como `?` y `*`) se utiliza como cadena de búsqueda. Se cargan los archivos XML cuyo nombre coincida con la expresión de búsqueda y sus nodos de documento se devuelven en forma de secuencia (*ver ejemplos más abajo*).
- Ejemplo XSLT: la expresión `collection("c:\MyDocs\*.xml")//Title` devuelve una secuencia de todos los elementos `DocTitle` del archivo `.xml` de la carpeta `MyDocs`.
- Ejemplo XQuery: la expresión `{for $i in collection(c:\MyDocs\*.xml) return element doc{base-uri($i)}}` devuelve los URI base de todos los archivos `.xml` de la carpeta `MyDocs`, donde cada URI está dentro de un elemento `doc`.
- La intercalación predeterminada está vacía.

---

current-date, current-dateTime, current-time

- La fecha y hora actual se toma del reloj del sistema.
- El uso horario se toma del uso horario implícito que se obtiene al evaluar el contexto. El uso horario implícito se toma del reloj del sistema.
- El uso horario siempre se especifica en el resultado.

---

doc

Solamente se emite un error si no hay ningún archivo XML disponible en la ubicación especificada o si el archivo no tiene un formato XML correcto. El archivo se valida si hay un esquema disponible. Si el archivo no es válido, el archivo no válido se carga sin información de esquema.

---

id

En un documento con formato XML correcto pero no válido que contiene dos o más elementos con el mismo valor ID, se devuelve el primer elemento del orden de documento.

---

in-scope-prefixes

En el documento XML solamente se puede anular la declaración de espacios de nombres predeterminados. Sin embargo, incluso al anular la declaración de un espacio de nombres predeterminado de un nodo de elemento, se devuelve el prefijo para el espacio de nombres predeterminado, que es una cadena de longitud cero.

---

normalize-unicode

Los formatos de normalización compatibles son NFC, NFD, NFKC y NFKD.

---

resolve-uri

- Si se omite el segundo argumento, opcional, el URI que se debe resolver (el primer argumento) se resuelve con el URI base del contexto estático, que es el URI de la hoja de estilos XSLT o el URI base dado en el prólogo del documento XQuery.
- El URI relativo (el primer argumento) se anexa después del último "/" de la notación de ruta de acceso de la notación del URI base.
- Si el valor del primer argumento es la cadena de longitud cero, se devuelve el URI base del contexto estático y este URI incluye el nombre de archivo del documento del que se deriva el URI base del contexto estático (p. ej. el archivo XSLT o XML).

---

static-base-uri

El URI base del contexto estático es el URI base de la hoja de estilos XSLT o el URI base especificado en el prólogo del documento XQuery.

 © 2013 Altova GmbH

8.7 Funciones XQuery y XPath 3.0

Las funciones y operadores XPath y XQuery 3.0 compatibles con RaptorXML (en todas las ediciones) cumplen con la recomendación del W3C sobre Funciones y operadores XPath y XQuery 3.0 del 21 de mayo de 2013.

Tienen las mismas características que las funciones XQuery 1.0 y XPath 2.0.

Altova RaptorXML 2013

Funciones de extensión XSLT y XQuery

9 Funciones de extensión XSLT y XQuery

En los lenguajes de programación, como Java y C#, hay varias funciones predefinidas que no están disponibles como funciones XQuery/XPath ni como funciones XSLT. Un buen ejemplo son las funciones matemáticas disponibles en Java, como `sin()` y `cos()`. Si estas funciones estuvieran a disposición de los diseñadores de hojas de estilos XSLT y consultas XQuery, el ámbito de aplicación de las hojas de estilos y de las consultas sería mayor y sería más fácil crearlas.

Los motores XSLT y XQuery utilizados en los productos de Altova admiten el uso de funciones de extensión en Java y .NET. También admiten el uso de scripts MSXSL para XSLT y las funciones de extensión de Altova.

Tenga en cuenta que, excepto algunas funciones de extensión de Altova para XSLT, a casi todas las funciones de extensión que aparecen en este anexo se les llama desde expresiones XPath. Este anexo describe cómo usar funciones de extensión y scripts MSXSL en sus hojas de estilos XSLT y en sus documentos XQuery. El anexo se divide en estos apartados:

- Funciones de extensión de Altova
- Funciones de extensión Java
- Funciones de extensión .NET
- Scripts MSXSL para XSLT

Los dos principales aspectos que se tienen en cuenta en estos apartados son: (i) cómo se llama a las funciones de cada biblioteca y (ii) qué reglas se siguen para convertir los argumentos de una llamada a función en el formato de entrada de la función y qué reglas se siguen para la conversión de retorno (el resultado de la función se convierte en objeto de datos XSLT/XQuery).

Requisitos
Para poder trabajar con funciones de extensión, en el equipo que ejecuta la transformación XSLT o la ejecución XQuery debe tener instalado (o debe tener acceso a) un entorno de ejecución Java (para acceder a funciones Java) y .NET Framework 2.0 (mínimo, para acceder a funciones .NET).

© 2013 Altova GmbH

9.1 Funciones de extensión de Altova

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

```
altova:
```

que se supone estará enlazado al espacio de nombres señalado.

Las funciones de extensión que aparecen más abajo son compatibles con la versión actual de su producto de Altova.

Funciones generales

Funciones XPath

Puede usar estas funciones en contextos XPath:

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

Funciones XSLT

Puede usar estas funciones en un contexto XSLT, igual que las funciones XSLT 2.0 `current-group()` o `key()`:

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

Funciones de gráficos (solamente en las ediciones Enterprise y Server)

Las funciones de extensión de Altova para gráficos solamente son compatibles con las ediciones Enterprise y Server de los productos de Altova y permiten generar gráficos a partir de datos XML. Hay dos tipos de funciones para gráficos:

- Funciones para generar y guardar gráficos
- Funciones para crear gráficos

En este apartado también incluimos un fragmento de código de la estructura XML de los datos de gráfico, a partir de la cual se pueden generar gráficos. Para terminar, ofrecemos un documento XSLT de muestra que ilustra cómo usar funciones para gráficos para generar gráficos a partir de datos XML.

9.1.1 Funciones generales

Esta son las funciones de extensión generales compatibles con la versión actual de su producto de Altova. Sin embargo, tenga en cuenta que algunas funciones pueden cambiar o dejar de ser compatibles en las próximas versiones del software. Remítase a la documentación de las versiones nuevas del software para consultar las funciones de extensión de Altova compatibles con la versión.

Recuerde que 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

```
altova:
```

que se supone estará enlazado al espacio de nombres señalado.

Funciones para usar en contextos XPath
Puede usar estas funciones en contextos XPath:

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

Funciones para usar en contextos XSLT
Puede usar estas funciones en un contexto XSLT, igual que las funciones XSLT 2.0 `current-group()` o `key()`:

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

Funciones para usar en contextos XPath
Puede usar estas funciones en contextos XPath:

altova:generate-auto-number(id as xs:string, start-with as xs:double, increment as xs:double, reset-on-change as xs:string)
Genera una serie de números que tienen el ID especificado. Se especifica el entero inicial y el incremento.

altova:reset-auto-number(id as xs:string)
Esta función restablece la numeración automática de la serie de numeración automática especificada con el argumento de Id. La serie vuelve a empezar por el entero inicial de la serie (ver la función `altova:generate-auto-number`).

 © 2013 Altova GmbH

altova:get-temp-folder as xs:string
Obtiene la carpeta temporal.

Funciones para usar en contextos XSLT

Puede usar estas funciones en un contexto XSLT, igual que las funciones XSLT 2.0 `current-group()` o `key()`:

altova:evaluate()
Toma una expresión XPath, pasada en forma de cadena, como argumento obligatorio. Devuelve el resultado de la expresión evaluada.

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

Por ejemplo:

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

Aquí, la expresión `//Name[1]` se pasa como cadena si se escribe entre comillas simples. La función `altova:evaluate` devuelve el contenido del primer elemento `Name` del documento.

La función `altova:evaluate` puede tomar otros argumentos (opcionales). Estos argumentos son los valores de las variables de nombre `p1`, `p2`, `p3`... `pN` que se pueden usar en la expresión XPath.

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

donde

- los nombres de variable deben tener el formato `pX`, siendo `X` un número entero
- la secuencia de los argumentos de la función, a partir del segundo argumento, corresponde a la secuencia de las variables `p1` a `pN`. De modo que el segundo argumento será el valor de la variable `p1`, el tercero será el valor de la variable `p2` y así sucesivamente.
- los valores de variable deben ser de tipo `item*`

Por ejemplo:

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

Resultado: "`hi 20 10`"

Observe que en el ejemplo anterior:

- 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.
- el cuarto argumento de la función es un valor de cadena, al estar escrito entre comillas simples.
- el atributo `select` del elemento `xs:variable` suministra la expresión XPath. Puesto que la expresión debe ser de tipo `xs:string`, se escribe entre comillas simples.

Aquí tiene más ejemplos:

```
<xsl:variable name="xpath" select="'$p1'" />
<xsl:value-of select="altova:evaluate( $xpath, //Name[1] )" />
```
Resultado: el valor del primer elemento `Name`.

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

En la función de extensión `altova:evaluate()` puede usar estas variables:

- Variables estáticas: `<xsl:value-of select="$i3, $i2, $i1" />`
  Resultado: el valor de las tres variables.

- Expresiones XPath dinámicas con variables dinámicas:
  ```
  <xsl:variable name="xpath" select="'$p3, $p2, $p1'" />
  <xsl:value-of select="altova:evaluate( $xpath, 10, 20, 30 )" />
  ```
  Resultado: "`30 20 10`*"*

- Expresiones XPath dinámicas sin variables dinámicas:
  ```
  <xsl:variable name="xpath" select="'$p3, $p2, $p1'" />
  <xsl:value-of select="altova:evaluate( $xpath )" />
  ```
  Resultado: "No variable defined for $p3."

Nota: el contexto estático incluye espacios de nombres, tipos y funciones (pero no variables) del entorno desde donde se llama a la función. El URI base y el espacio de nombres predeterminado se heredan.

`altova:distinct-nodes()`

La función `altova:distinct-nodes()` toma un conjunto de nodos como entrada y devuelve el mismo conjunto menos los nodos que tienen el mismo valor. La comparación se realiza por medio de la función XPath/XQuery `fn:deep-equal`.

 © 2013 Altova GmbH

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

altova:encode-for-rtf()

La función `altova:encode-for-rtf()` convierte la cadena de entrada en código para RTF.

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

Los espacios en blanco y las líneas nuevas se conservan dependiendo del valor booleano especificado en sus respectivos parámetros.

altova:xbrl-labels()

La función `altova:xbrl-labels()` toma dos argumentos de entrada: un nombre de nodo y la ubicación del archivo de taxonomía que incluye el nodo. La función devuelve las etiquetas XBRL asociadas al nodo de entrada.

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

altova:xbrl-footnotes()

La función `altova:footnotes()` toma un nodo como argumento de entrada y devuelve el conjunto de los nodos de nota al pie XBRL a los que hace referencia el nodo de entrada.

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

9.1.2 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 `C:\Archivos de programa\Altova\CommonYYYY\jar`.

Las bibliotecas Java utilizadas están en las subcarpetas de la carpeta `C:\Archivos de programa\Altova\CommonYYYY\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.

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`.

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:

`class` **`QRCodeBean`**

- *Extiende* `org.krysalis.barcode4j.impl.AbstractBarcodeBean`
- *Crea una interfaz* `AbstractBarcodeBean` *para* `com.google.zxing.qrcode.encoder`

 © 2013 Altova GmbH

```
    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
```

Plantilla XSLT

A continuación puede ver un ejemplo de plantilla 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:generateBarcodePngAsHexS
tring($barcodeObject)) ))"/>
      </xsl:result-document>
   </xsl:template>
</xsl:stylesheet>
```

9.1.3 Funciones 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. Sin embargo, tenga en cuenta que algunas funciones pueden cambiar o dejar de ser compatibles en las próximas versiones del software. Remítase a la documentación de las versiones nuevas del software para consultar las funciones de extensión de Altova compatibles con la versión.

Hay dos tipos de funciones para gráficos:

- Funciones para generar y guardar gráficos
- Funciones para crear gráficos

Nota: las funciones para gráficos solamente son compatibles con los **productos servidor** de Altova y con las ediciones **Enterprise Edition** de las herramientas de escritorio de Altova.

Nota: en los **productos servidor** de Altova los formatos de imagen compatibles para los 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 los 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

- `$chart` es el componente de extensión de gráficos obtenido con la función `altova:create-chart`
- `$width` y `$height` deben especificarse con una unidad de longitud
- `$encoding` puede ser `binarytobase64` o `binarytobase16`

La función devuelve la imagen del gráfico en la codificación elegida.

```
altova:generate-chart-image ($chart, $width, $height, $encoding, $imagetype)
as atomic
```

donde

- `$chart` es el componente de extensión de gráficos obtenido con la función `altova:create-chart`
- `$width` y `$height` deben especificarse con una unidad de longitud
- `$encoding` puede ser `binarytobase64` o `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.

 © 2013 Altova GmbH

altova:save-chart-image (`$chart, $filename, $width, $height) as empty()` *(solo en Windows)*

donde

- `$chart` es el componente de extensión de gráficos obtenido con la función `altova:create-chart`
- `$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`.

altova:save-chart-image (`$chart, $filename, $width, $height, $imagetype) as empty()` *(solo en Windows)*

donde

- `$chart` es el componente de extensión de gráficos obtenido con la función `altova:create-chart`
- `$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
- `$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.

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

- `$chart-config` es el componente de extensión chart-config obtenido con la función `altova:create-chart-config` o con la función `altova:create-chart-config-from-xml`
- `$chart-data-series` es el componente de extensión chart-data-series obtenido con la función `altova:create-chart-data-series` o con la función `altova: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:create-chart-config(`$type-name, $title)` *como componente de extensión* `chart-config`

donde

- `$type-name` especifica el tipo de gráfico que se debe 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* `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.

`altova:create-chart-data-series``($series-name?, $x-values*, $y-values*)` *como componente de extensión* `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* `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* `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

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* `chart-layer`

 © 2013 Altova GmbH

donde

- `$chart-config` es el componente de extensión chart-config obtenido con la función `altova:create-chart-config` o con la función `altova:create-chart-config-from-xml`
- `$chart-data-series` es el componente de extensión chart-data-series que se obtiene con la función `altova:create-chart-data-series` o con la función `altova: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

- `$chart-config` es el componente de extensión chart-config obtenido con la función `altova:create-chart-config` o con la función `altova:create-chart-config-from-xml`
- `$chart-data-series` es el componente de extensión chart-data-series obtenido con la función `altova:create-chart-data-series` o con la función `altova:create-chart-data-series-from-rows`
- `$chart-layer` es el componente de extensión chart-layer obtenido con la función `altova:create-chart-layer`

La función devuelve un componente de extensión multi-layer-chart.

`altova:create-multi-layer-chart``($chart-config, $chart-data-series*, $chart-layer*, xs:boolean $mergecategoryvalues)`

donde

- `$chart-config` es el componente de extensión chart-config obtenido con la función `altova:create-chart-config` o con la función `altova:create-chart-config-from-xml`
- `$chart-data-series` es el componente de extensión chart-data-series obtenido con la función `altova:create-chart-data-series` o con la función `altova:create-chart-data-series-from-rows`
- `$chart-layer` es el componente de extensión chart-layer obtenido con la función `altova:create-chart-layer`

La función devuelve un componente de extensión multi-layer-chart.

9.1.4 Estructura XML de los datos de gráfico

A continuación puede ver una estructura XML de los datos de gráfico, tal y como aparecería para las funciones de extensión para gráficos de Altova. 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"
         MinFontHeight="10.pt"
         Size="5.%" />
   </General>
```

 © 2013 Altova GmbH

<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"
```

 © 2013 Altova GmbH

```
            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.%"
        />
        <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>
```

 © 2013 Altova GmbH

9.1.5 Ejemplo: funciones de gráficos

A continuación puede ver un ejemplo de documento XSLT que muestra cómo usar las funciones de extensión para gráficos de Altova. Después ofrecemos un documento XML y una captura de pantalla de la imagen generada cuando el documento XML se procesa con el documento XSTL usando el motor XSLT 2.0 o 3.0.

Nota: las funciones para gráficos son compatibles solamente con las ediciones **Enterprise** y **Server Editions** de los productos de Altova.

Nota: para más información sobre las tablas de datos del gráfico, consulte la documentación de las herramientas de Altova XMLSpy y StyleVision.

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 XSLT 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>
```

Documento XML

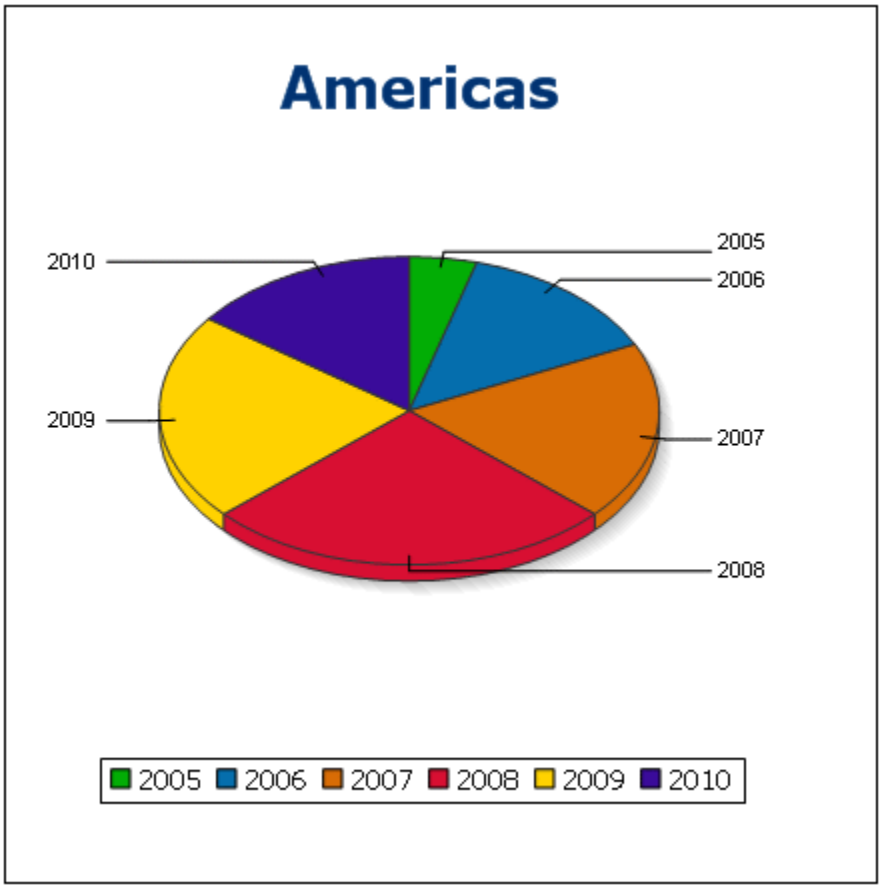
Este documento XML se puede procesar con el XSLT que aparece arriba. Los datos del documento XML se usan para generar el gráfico circular de la imagen que aparece más abajo.

```
<?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

 © 2013 Altova GmbH

Este gráfico circular se generó al procesar el documento XML anterior con el archivo XSLT.



9.2 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:

- Java: constructores
- Java: métodos estáticos y campos estáticos
- Java: métodos de instancia y campos de instancia
- Tipos de datos: XSLT/XQuery a Java
- Tipos de datos: Java a XSLT/XQuery

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.

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:

 © 2013 Altova GmbH

```
<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 y Archivos Jar definidos por el usuario. 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.

9.2.1 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 más abajo*.)
- 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 más abajo*.)
- El archivo de clases está en un paquete. El archivo XSLT/XQuery está en una carpeta cualquiera. (*Ver ejemplo más abajo*.)
- El archivo de clases no está en un paquete. El archivo XSLT/XQuery está una carpeta cualquiera. (*Ver ejemplo más abajo*.)

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 no está en un paquete. 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` del paquete `com.altova.extfunc`. El archivo de clases `Car` está en esta carpeta: `JavaProject/com/altova/extfunc`. El archivo XSLT también está en la carpeta

 © 2013 Altova GmbH

JavaProject/com/altova/extfunc.

```
<xsl:stylesheet version="2.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      xmlns:xs="http://www.w3.org/2001/XMLSchema"
      xmlns:fn="http://www.w3.org/2005/xpath-functions"
      xmlns:car="java:Car" >
<xsl:output exclude-result-prefixes="fn car xsl fo xs"/>

<xsl:template match="/">
   <a>
   <xsl:value-of select="car:getVehicleType()"/>
   </a>
</xsl:template>

</xsl:stylesheet>
```

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` 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-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.

 © 2013 Altova GmbH

9.2.2 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 preifjo 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
`!/` 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.

9.2.3 Java: 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, 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*):
```
<xsl:value-of select="date:toString(date:new())" xmlns:date="
java:java.util.Date" />
```

 © 2013 Altova GmbH

9.2.4 Java: 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>
```

9.2.5 Java: 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`.

 © 2013 Altova GmbH

9.2.6 Tipos de datos: XPath/XQuery a 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.

9.2.7 Tipos de datos: Java a 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.

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).

 © 2013 Altova GmbH

9.3 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:

- .NET: constructores
- .NET: métodos estáticos y campos estáticos
- .NET: métodos de instancia y campos de instancia
- Tipos de datos: XSLT/XQuery en .NET
- Tipos de datos: .NET en XSLT/XQuery

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"
   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

 © 2013 Altova GmbH

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.

9.3.1 .NET: 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, 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*):
```
<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(data:get_Month(date:new(2008, 4, 29))
" xmlns:date="clitype:System.DateTime" />
```

 © 2013 Altova GmbH

9.3.2 .NET: métodos 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"/>
```

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>
```

9.3.3 .NET: 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.

 © 2013 Altova GmbH

9.3.4 Tipos de datos: XPath/XQuery a .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`.
- 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.

9.3.5 Tipos de datos: .NET a 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).

 © 2013 Altova GmbH

9.4 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>
  </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" />
```

 © 2013 Altova GmbH

```
    ...
</msxsl:script>
```

El valor del atributo `namespace` es el nombre del espacio de nombres.

Altova RaptorXML 2013

Altova LicenseServer

10 Altova LicenseServer

Altova LicenseServer (en adelante *LicenseServer*) ofrece un lugar común donde administrar todas las licencias de los productos de Altova. Las aplicaciones de Altova que se ejecuten en la red pueden tener asignadas licencias desde el servidor LicenseServer. Esto permite al administrador gran flexibilidad a la hora de gestionar y supervisar licencias.

Proceso de asignación de licencias con Altova LicenseServer
Para asignar una licencia a un producto servidor de Altova con Altova LicenseServer:

1. Inicie LicenseServer.
2. Abra la página de configuración de LicenseServer, es decir, la interfaz de administración de LicenseServer (en Windows o Linux).
3. Cargue las licencias que Altova le envió. Esto se hace en la pestaña **License Pool** de la página de configuración de LicenseServer.
4. Registre el producto servidor de Altova con LicenseServer (FlowForce Server, MapForce Server, StyleVision Server, Registrar RaptorXML(+XBRL) Server con LicenseServer). El proceso de registro depende del tipo de producto: puede ser desde la interfaz gráfica del producto o desde su línea de comandos. Para más información sobre cómo registrar su producto servidor con LicenseServer consulte la documentación de cada producto.
5. En la pestaña **Server Management** de la página de configuración de LicenseServer, asigne una licencia al producto servidor en función del número de núcleos del equipo en el que se ejecuta el producto.

A partir de ese momento el administrador puede supervisar y administrar las licencias desde LicenseServer. Para más información consulte la *Referencia de la página de configuración*.

Nota sobre la documentación
La presente documentación se divide en tres secciones:

- Información general sobre requisitos de red, proceso de instalación en Windows y Linux y descripción de Altova ServiceController.
- Asignación de licencias: describe paso a paso el proceso de asignación de licencias en Altova LicenseServer.
- Referencia de la página de configuración: describe la interfaz de administración de LicenseServer y sus funciones.

Nota: la interfaz de administración de LicenseServer no es compatible con SSL.

Última actualización: 23/07/2013

© 2013 Altova GmbH

10.1 Información de red

Todos los equipos clientes que ejecutan productos servidor de Altova que necesitan una licencia deben tener acceso a un equipo servidor en el que debe estar instalado Altova LicenseServer. Los servidores de seguridad o firewalls (tanto del servidor como del cliente) deben permitir el tráfico de red con dicho servidor LicenseServer.

En LicenseServer, el **puerto 35355** se utiliza para distribuir licencias y, por tanto, debe estar abierto al tráfico de red con los equipos clientes.

Estos son los parámetros de red predeterminados y los requisitos de LicenseServer:

- *Para distribución de licencias con LicenseServer:*
  Uno de estos dos:
  Conexión TCP IPv4 en el puerto `35355`
  Conexión TCP IPv6 en el puerto `35355`

Para trabajar en tareas administrativas el servidor LicenseServer ofrece una interfaz web que utiliza el puerto 8088. El usuario puede configurar el puerto utilizado para la interfaz web según sus requisitos.

Conexión con el servidor maestro de licencias altova.com
Altova LicenseServer necesita comunicarse con el servidor maestro de licencias `altova.com` para validar y verificar los datos de licencia y para asegurarse de que los contratos de licencia de Altova se cumplen en todo momento. Esta comunicación se establece por HTTPS a través del puerto 443. Si tras completar la verificación inicial con el servidor maestro `altova.com`, Altova LicenseServer no puede volver a conectarse con `altova.com` durante más de 5 días (120 horas), se bloqueará el uso de los productos de Altova que estén conectados con Altova LicenseServer.

Cuando se pierde la conexión con el servidor maestro `altova.com`, la incidencia se registra en la pestaña **Messages** de la página de configuración de Altova LicenseServer. Además el administrador puede configurar Altova LicenseServer para que le envíe correos de alerta cuando se pierda la conexión con `altova.com`. Las opciones de configuración de estos correos de alerta se pueden definir en la pestaña **Settings** de la página de configuración.

10.2 Instalación en Windows

En los sistemas Windows puede instalar Altova LicenseServer de dos maneras:

- como producto servidor independiente.
- con el paquete de instalación de un producto servidor de Altova (Altova FlowForce Server, Altova MapForce Server o Altova SyleVision Server).

Si LicenseServer no está instalado, la opción *Instalar LicenseServer* se activa por defecto durante el proceso de instalación del producto servidor de Altova. Si LicenseServer ya está instalado, la opción *Instalar LicenseServer* se desactiva por defecto. Puede activar o desactivar la opción si quiere.

Nota: si quiere volver a instalar Altova LicenseServer, primero es necesario desinstalar la versión antigua.

Para aprender a asignar licencias con Altova LicenseServer consulte la sección Asignación de licencias.

 © 2013 Altova GmbH

10.3 Instalación en Linux

Puede instalar LicenseServer en sistemas Linux (Debian, Ubuntu, CentOS y RedHat).

Desinstalar versiones anteriores de LicenseServer

En la interfaz de la línea de comandos de Linux, compruebe si LicenseServer está instalado ya. Para ello use este comando:

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

Si LicenseServer no está instalado, instale el producto tal y como se explica más abajo. Si LicenseServer ya está instalado y desea instalar una versión más reciente, desinstale la versión antigua con este comando:

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

Instalar Altova LicenseServer

En los sistemas Linux debe instalar LicenseServer por separado porque no viene incluido en los paquetes de instalación de los productos servidor de Altova. Descargue el instalador de Altova LicenseServer del sitio web de Altova y copie el paquete en cualquier directorio del sistema Linux.

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

En una ventana de la Terminal, cambie al directorio donde copió el paquete de 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 licenseserver-1.1-debian.deb
[Ubuntu]:   sudo dpkg --install licenseserver-1.1-ubuntu.deb
[CentOS, RedHat]:   sudo rpm -ivh licenseserver-1.1-1.x86_64.rpm
```

El paquete de LicenseServer se instalará en la carpeta:

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

Para aprender a asignar licencias con Altova LicenseServer consulte la sección Asignación de licencias.

10.4 Instalación en Mac OS X

Puede instalar Altova LicenseServer en sistemas Mac OS X (versión 10.7 o superior). En este apartado describimos primero el proceso de desinstalación, por si necesita desinstalar una versión previa de LicenseServer.

Desinstalar versiones previas de LicenseServer
Antes de desinstalar LicenseServer debe detener el servicio con este comando:

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

Para comprobar si el servicio se detuvo o no, abra la terminal del Monitor de actividad y confirme que LicenseServer no está en la lista.

En la terminal de Aplicaciones haga clic con el botón secundario en el icono de LicenseServer y seleccione **Mover a la papelera**. La aplicación se envía a la papelera pero debe quitar la aplicación de la carpeta `usr`. Para ello puede utilizar este comando:

```
sudo rm -rf /usr/local/Altova/LicenseServer11
```

Si necesita desinstalar una versión antigua de Altova LicenseServer, utilice el procedimiento descrito para FlowForce Server.

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

Instalar Altova LicenseServer
Descargue Altova LicenseServer del sitio web de Altova (el archivo del instalador tiene la extensión `.pkg`) y haga doble clic en el paquete del instalador para iniciar la instalación. Siga los pasos del asistente para la instalación y acepte el contrato de licencia.

Altova LicenseServer se instala en esta carpeta:

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

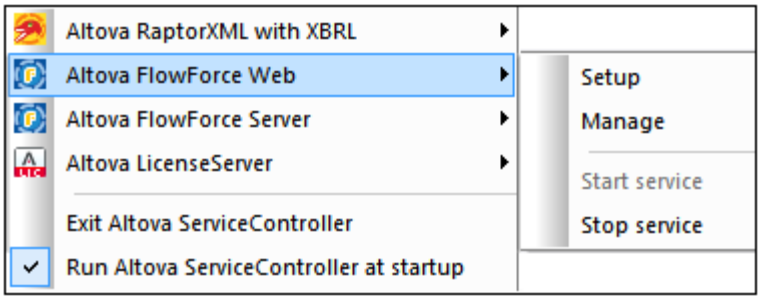

© 2013 Altova GmbH

10.5 Altova ServiceController

El controlador de servicios Altova ServiceController (en adelante *ServiceController*) es una práctica aplicación con la que podrá iniciar, detener y configurar los servicios de Altova en **sistemas Windows**. Este controlador no está disponible en sistemas Linux.

Altova ServiceController se instala automáticamente con Altova LicenseServer y se puede iniciar haciendo clic en **Inicio | Altova LicenseServer | Altova ServiceController**. Una vez iniciado, podrá acceder al controlador desde la bandeja del sistema (*imagen siguiente*).

Para ejecutar ServiceController tras iniciar sesión en el sistema, haga clic en su icono de la bandeja del sistema. Aparece un menú contextual (*imagen siguiente*) donde puede activar el comando **Run Altova ServiceController at Startup** para que Altova ServiceController se ejecute nada más iniciar el sistema. Este comando está activado por defecto. Para cerrar ServiceController, haga clic en el icono de la bandeja del sistema y en el menú contextual (*imagen siguiente*) seleccione **Exit** para cerrar Altova ServiceController.



Iniciar y detener servicios de Altova
Todos los servicios de Altova instalados en el equipo aparecen en el menú contextual de ServiceController (*imagen anterior*). Los servicios de Altova se pueden iniciar o detener con los comandos del menú contextual de ServiceController. Este menú también incluye comandos de algunas tareas de administración. Por ejemplo, en la imagen anterior puede ver el submenú del servicio Altova FlowForce Web, donde puede elegir la opción **Setup** para abrir la página de configuración de Altova FlowForce Server Web.

10.6 Asignación de licencias

Siga estos pasos para asignar una licencia a un producto servidor de Altova desde Altova LicenseServer:

1. Inicie LicenseServer.
2. Abra la página de configuración de LicenseServer, es decir, la interfaz de administración de LicenseServer (en Windows o Linux).
3. Cargue las licencias que Altova le envió. Esto se hace en la pestaña **License Pool** (Repertorio de licencias) de la página de configuración de LicenseServer.
4. Registre el producto servidor de Altova (FlowForce Server, MapForce Server, StyleVision Server) con LicenseServer. El proceso de registro depende del tipo de producto: puede ser desde la interfaz gráfica del producto o desde su línea de comandos. Para más información sobre cómo registrar su producto servidor con LicenseServer consulte la documentación de cada producto.
5. En la pestaña **Server Management** (Gestión de servidores) de la página de configuración de LicenseServer, asigne una licencia al producto servidor en función del número de núcleos del equipo en el que se ejecuta el producto.

Nota sobre núcleos y licencias
La asignación de licencias a productos servidor de Altova depende de cuántos núcleos de procesador tiene el 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 licencia asignada a un producto instalado en un equipo servidor debe ser mayor o igual al número de núcleos disponibles en dicho servidor, ya sea un equipo virtual o uno físico.

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 será menos rápida que si utilizara todos los núcleos disponibles en el servidor.

> **Nota:** cada licencia se puede usar solamente para un equipo cliente, incluso si no se usa la totalidad de la licencia. 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 para otro equipo cliente simultáneamente.

© 2013 Altova GmbH

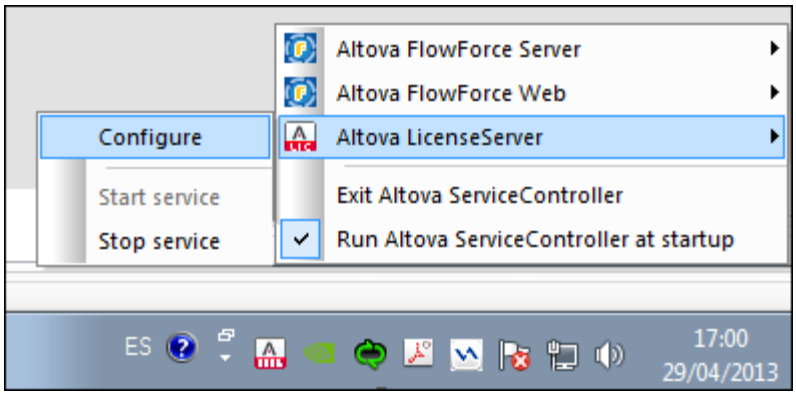
10.6.1 Iniciar LicenseServer

Este apartado explica:

- Cómo iniciar LicenseServer en sistemas Windows
- Cómo iniciar LicenseServer en sistemas Linux
- Cómo iniciar LicenseServer en sistemas Mac OS X
- La conexión con el servidor maestro altova.com

Sistemas Windows
Puede iniciar LicenseServer con ayuda de ServiceController, disponible en la bandeja del sistema. (Haga clic en **Inicio | Todos los programas | Altova LicenseServer | Altova ServiceController** para iniciar Altova ServiceController, que a partir de ese momento estará disponible en la bandeja del sistema. Si selecciona la opción **Run Altova ServiceController at Startup** (*imagen siguiente*), Altova ServiceController se iniciará nada más iniciar sesión en el sistema y su icono aparecerá en la bandeja del sistema.

Para iniciar LicenseServer, haga clic en el icono de Altova ServiceController de la bandeja del sistema, pase el puntero del mouse sobre el submenú **Altova LicenseServer** (*imagen siguiente*) y seleccione **Start Service** para iniciar el servicio. Si LicenseServer ya está en ejecución, la opción **Start Service** aparece deshabilitada (*imagen siguiente*).



Sistemas Linux
Para iniciar LicenseServer como servicio en sistemas Linux, ejecute este comando en una ventana de la Terminal.

```
[Debian]:    sudo /etc/init.d/licenseserver start

[Ubuntu]:    sudo initctl start licenseserver

[CentOS, RedHat]:    sudo initctl start licenseserver
```

(Si necesita detener LicenseServer, reemplace `start` por `stop` en el comando anterior.)

Sistemas Mac OS X
Para iniciar LicenseServer como servicio en sistemas Mac OS X, ejecute este comando en una ventana de la Terminal:

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

Si necesita detener LicenseServer por cualquier motivo, use este comando:

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

Conexión con el servidor maestro de licencias altova.com
Altova LicenseServer necesita comunicarse con el servidor maestro de licencias `altova.com` para validar y verificar los datos de licencia y para asegurarse de que los contratos de licencia de Altova se cumplen en todo momento. Esta comunicación se establece por HTTPS a través del puerto 443. Si tras completar la verificación inicial con el servidor maestro `altova.com`, Altova LicenseServer no puede volver a conectarse con `altova.com` durante más de 5 días (120 horas), se bloqueará el uso de los productos de Altova que estén conectados con Altova LicenseServer.

Cuando se pierde la conexión con el servidor maestro `altova.com`, la incidencia se registra en la pestaña **Messages** de la página de configuración de Altova LicenseServer. Además el administrador puede configurar Altova LicenseServer para que le envíe correos de alerta cuando se pierda la conexión con `altova.com`. Las opciones de configuración de estos correos de alerta se pueden definir en la pestaña **Settings** de la página de configuración.

© 2013 Altova GmbH

10.6.2 Abrir la página de configuración de LicenseServer (Windows)

Este apartado explica:

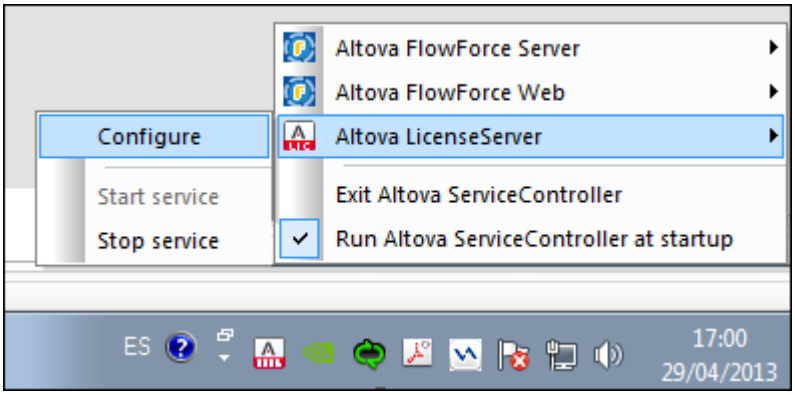
- Cómo abrir la página de configuración cuando LicenseServer está ejecutándose en el mismo equipo
- Cómo abrir la página de configuración cuando LicenseServer está ejecutándose en otro equipo
- Cómo iniciar sesión con la contraseña inicial
- Cómo establecer un puerto fijo para la página de configuración

---

Abrir la página de configuración cuando LicenseServer está ejecutándose en el mismo equipo
En los sistemas Windows, si el servidor LicenseServer está ejecutándose en el mismo equipo, puede abrir la página de configuración de LicenseServer de dos maneras:

- Haga clic en **Inicio | Todos los programas | Altova LicenseServer | Página de configuración de LicenseServer**. La página de configuración se abre en una pestaña nueva del explorador web.

- Haga clic en el icono de Altova ServiceController de la bandeja del sistema, pase el puntero del mouse sobre **Altova LicenseServer** en el menú emergente (*imagen siguiente*) y seleccione **Configure**.



La página de configuración se abre en una ventana del explorador nueva por la página de inicio de sesión (*imagen siguiente*).

Abrir la página de configuración cuando LicenseServer está ejecutándose en otro equipo
Para abrir la página de configuración de LicenseServer desde otro equipo Windows de la red local (cuando LicenseServer no está instalado en el equipo local), introduzca la URL de la página de configuración de LicenseServer en la barra de dirección del explorador web y pulse **Entrar**. La URL predeterminada de la página de configuración es:

```
http://<DirecciónIPoNombreDelServidor>:8088/
```

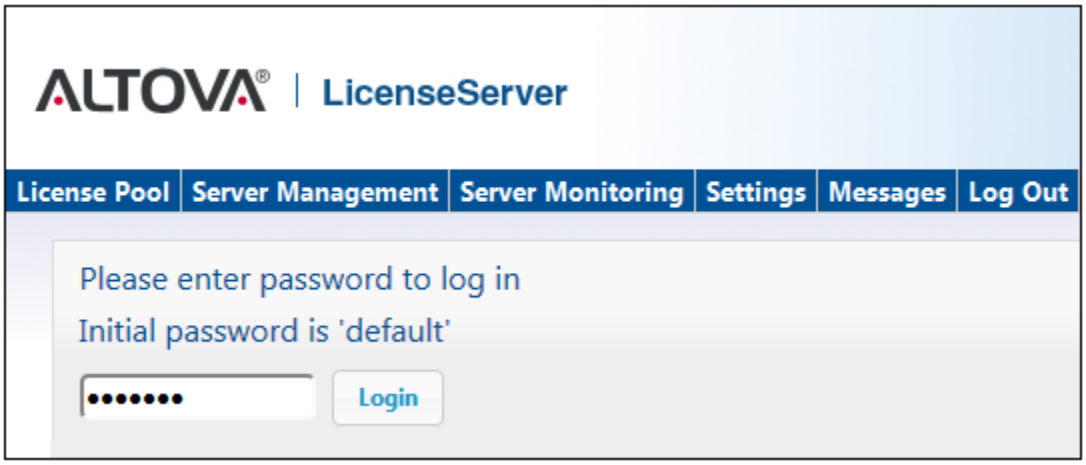
La URL de la página de configuración está presente en el código HTML de la página de configuración propiamente dicha, que se llama `WebUI.html` y está en este directorio:

```
C:/ProgramData/Altova/LicenseServer/WebUI.html
```

Si eligió que la URL de la página de configuración se genere de forma dinámica (en la pestaña **Configuración** de la página de configuración), la URL nueva se genera cada vez que se inicia LicenseServer. Para averiguar la URL actual de la página de configuración necesitará ver la versión actual de `WebUI.html`.

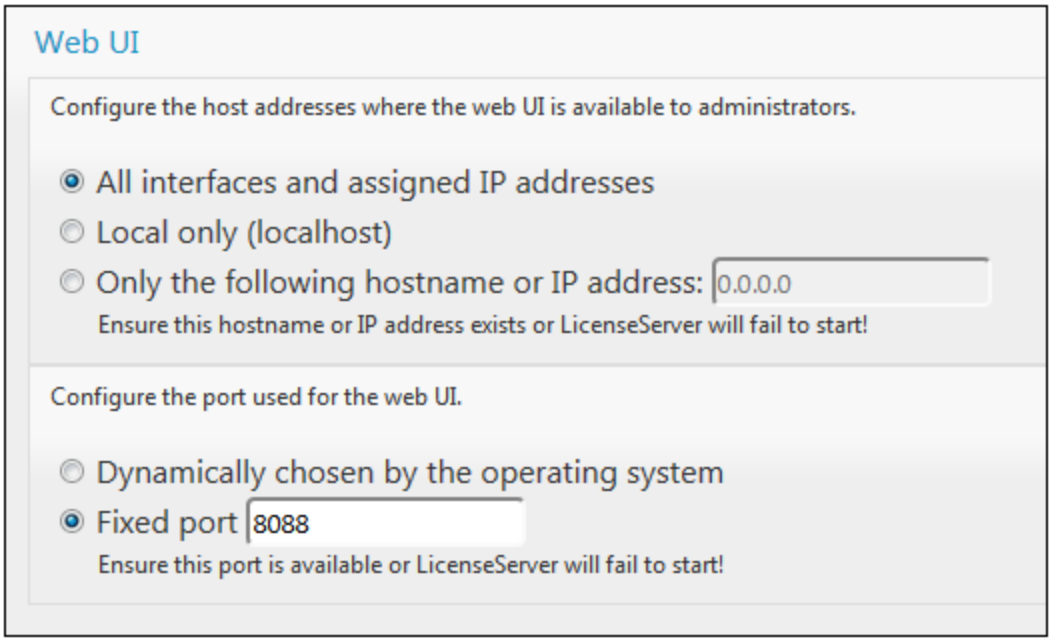
El formato de la URL generada dinámicamente en `WebUI.html` será similar a `http://127.0.0.1:55541/con-una-cadena-adicional-si-quiere` y se encuentra dentro de la función `checkIfServiceRunning()` en un script situado cerca del final del elemento `<head>`. El número de puerto de la URL se asigna dinámicamente y la parte de la dirección IP identifica el servidor en el que está instalado LicenseServer. Si quiere acceder a la página de configuración de LicenseServer desde otro equipo, compruebe que la dirección IP de la URL es correcta o incluye el nombre del servidor en el que está instalado LicenseServer. Por ejemplo: `http://MiServidor:55541`.

Iniciar sesión con la contraseña inicial
Después de seguir todas estas instrucciones se abre la página de configuración por la página de inicio de sesión (*imagen siguiente*). Puede iniciar sesión con la contraseña predeterminada `default`. Después de iniciar sesión puede cambiar la contraseña en la pestaña **Settings**.



Establecer un puerto fijo o dinámico para la página de configuración
El puerto de la página de configuración (y, por consiguiente, su dirección) se puede especificar en la pestaña **Settings**. El puerto predeterminado es `8088` pero puede definir cualquier puerto para la página de configuración de LicenseServer (*imagen siguiente*). También puede permitir que el puerto se seleccione dinámicamente cada vez que se inicie LicenseServer. En este caso, necesitará buscar la URL de la página de configuración en el archivo `WebUI.html` (consulte los apartados Abrir la página de configuración de LicenseServer (Windows) y Abrir la página de configuración de LicenseServer (Linux)).

 © 2013 Altova GmbH



La ventaja de usar un puerto fijo es que la URL de la página se conoce por adelantado y se puede abrir con más facilidad. Si el puerto se asigna dinámicamente, la parte del puerto de la URL debe buscarse en el archivo `WebUI.html` cada vez que se inicia LicenseServer.

10.6.3 Abrir la página de configuración de LicenseServer (Linux)

Este apartado explica:

- Cómo abrir por primera vez la página de configuración usando la URL obtenida
- La URL de la página de configuración de LicenseServer
- Cómo iniciar sesión con la contraseña inicial
- Cómo establecer un puerto fijo para la página de configuración

---

Abrir por primera vez la página de configuración usando la URL obtenida
Cuando se registra el producto servidor de Altova con LicenseServer por la interfaz de la línea de comandos en sistemas Linux, se devuelve la URL de la página de configuración de LicenseServer. Al abrir esta URL en el explorador web, se le solicita que lea y acepte el contrato de licencia. Después de aceptarlo, aparece la página de inicio de sesión de la página de configuración (*imagen siguiente*).

URL de la página de configuración de LicenseServer
Después, cuando quiera abrir la página de configuración, escriba la URL en la barra de dirección del explorador web y pulse **Entrar**. La URL predeterminada de la página de configuración es:

```
http://<DirecciónIPoNombreDelServidor>:8088/
```

La URL de la página de configuración está presente en el código HTML de la página de configuración propiamente dicha, que se llama **`WebUI.html`** y está en este directorio:

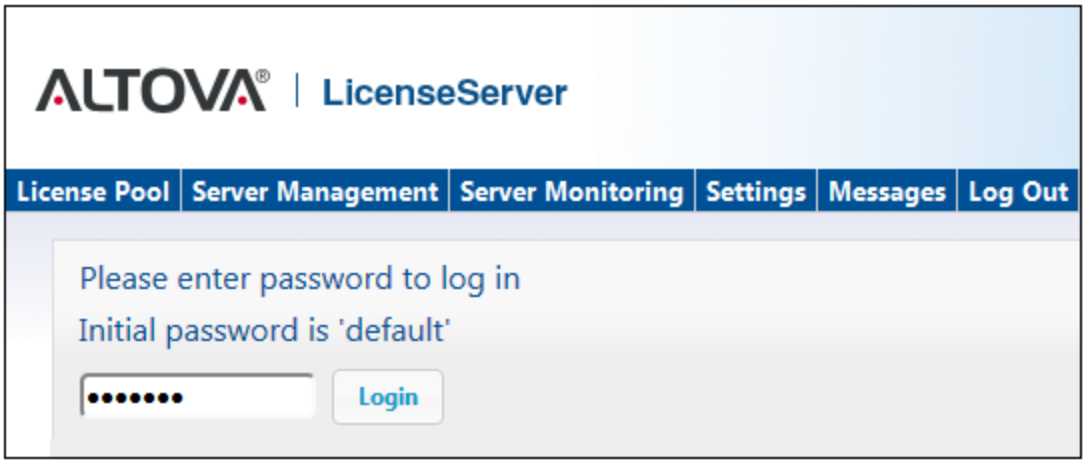
```
/var/opt/Altova/LicenseServer/webUI.html
```

Si eligió que la URL de la página de configuración se genere de forma dinámica (en la pestaña **Configuración** de la página de configuración), se genera una URL nueva cada vez que se inicia LicenseServer. Para averiguar la URL actual de la página de configuración necesitará mirar la versión actual de **`WebUI.html`**.

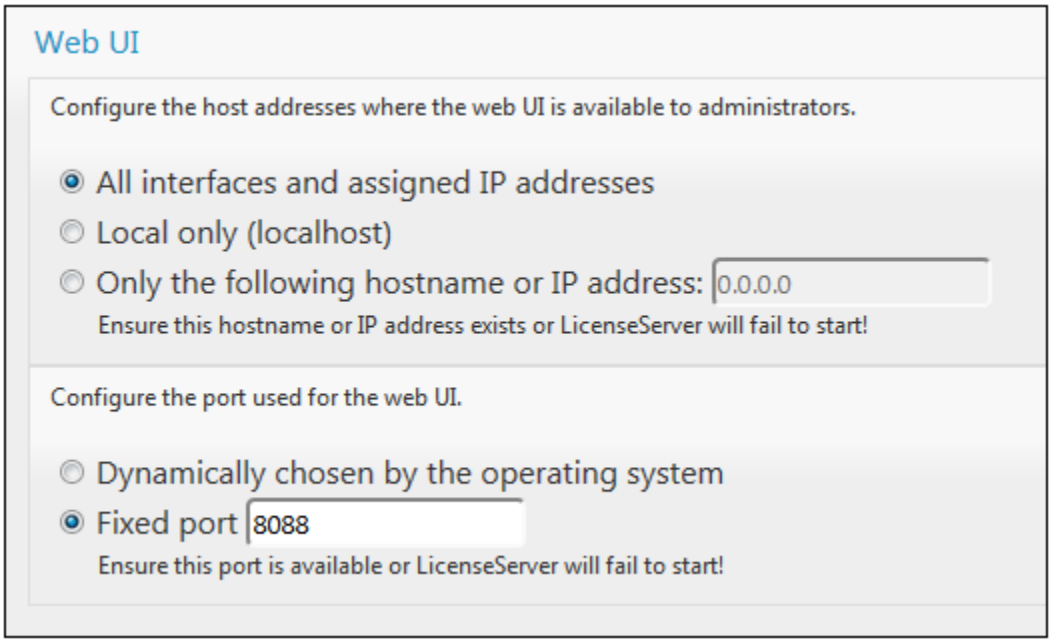
El formato de la URL generada dinámicamente en **`WebUI.html`** será similar a `http://127.0.0.1:55541` y se encuentra dentro de la función `checkIfServiceRunning()` en un script situado cerca del final del elemento `<head>`. El número de puerto de la URL se asigna dinámicamente y la parte de la dirección IP identifica el servidor en el que está instalado LicenseServer. Si quiere acceder a la página de configuración de LicenseServer desde otro equipo, compruebe que la dirección IP de la URL es correcta o incluye el nombre del servidor en el que está instalado LicenseServer. Por ejemplo: `http://MiServidor:55541`.

Iniciar sesión con la contraseña inicial
Después de seguir todas estas instrucciones se abre la página de configuración por la página de inicio de sesión (*imagen siguiente*). Puede iniciar sesión con la contraseña predeterminada `default`. Después de iniciar sesión puede cambiar la contraseña en la pestaña **Settings**.

 © 2013 Altova GmbH



Establecer un puerto fijo o dinámico para la página de configuración
El puerto de la página de configuración (y, por consiguiente, su dirección) se puede especificar en la pestaña **Settings**. El puerto predeterminado es `8088` pero puede definir cualquier puerto para la página de configuración de LicenseServer (*imagen siguiente*). También puede permitir que el puerto se seleccione dinámicamente cada vez que se inicie LicenseServer. En este caso, necesitará buscar la URL de la página de configuración en el archivo `WebUI.html` (consulte los apartados Abrir la página de configuración de LicenseServer (Windows) y Abrir la página de configuración de LicenseServer (Linux)).



La ventaja de usar un puerto fijo es que la URL de la página se conoce por adelantado y se puede abrir con más facilidad. Si el puerto se asigna dinámicamente, la parte del puerto de la URL debe buscarse en el archivo `WebUI.html` cada vez que se inicia LicenseServer.

10.6.4 Abrir la página de configuración de LicenseServer (Mac OS X)

Este apartado explica:

- Cómo abrir la página de configuración por primera vez usando la URL obtenida
- La URL de la página de configuración de LicenseServer
- Cómo iniciar sesión con la contraseña inicial
- Cómo establecer un puerto fijo para la página de configuración

---

Abrir por primera vez la página de configuración usando la URL obtenida
Cuando se registra el producto servidor de Altova con LicenseServer por la interfaz de la línea de comandos en sistemas Mac OS X, se devuelve la URL de la página de configuración de LicenseServer. Al abrir esta URL en el explorador web, se le solicita que lea y acepte el contrato de licencia. Después de aceptarlo, aparece la página de inicio de sesión de la página de configuración (*imagen siguiente*).

URL de la página de configuración de LicenseServer
Después, cuando quiera abrir la página de configuración, escriba la URL en la barra de dirección del explorador web y pulse **Entrar**. La URL predeterminada de la página de configuración es:

```
http://<DirecciónIPoNombreDelServidor>:8088/
```

La URL está presente en el código HTML de la página de configuración propiamente dicha, que se llama `WebUI.html` y está en este directorio:

```
/var/Altova/LicenseServer/webUI.html
```

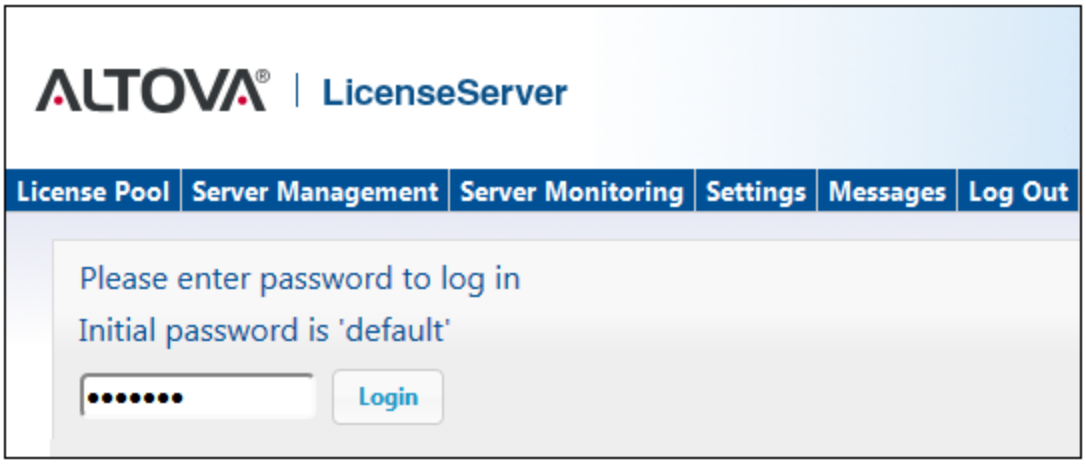
Si eligió que la URL de la página de configuración se genere de forma dinámica (en la pestaña **Configuración** de la página de configuración), se genera una URL nueva cada vez que se inicia LicenseServer. Para averiguar la URL actual de la página de configuración necesitará mirar la versión actual de `WebUI.html`.

El formato de la URL generada dinámicamente en `WebUI.html` será similar a `http://127.0.0.1:55541` y se encuentra dentro de la función `checkIfServiceRunning()` en un script situado cerca del final del elemento `<head>`. El número de puerto de la URL se asigna dinámicamente y la parte de la dirección IP identifica el servidor en el que está instalado LicenseServer. Si quiere acceder a la página de configuración de LicenseServer desde otro equipo, compruebe que la dirección IP de la URL es correcta o incluye el nombre del servidor en el que está instalado LicenseServer. Por ejemplo: `http://MiServidor:55541`.

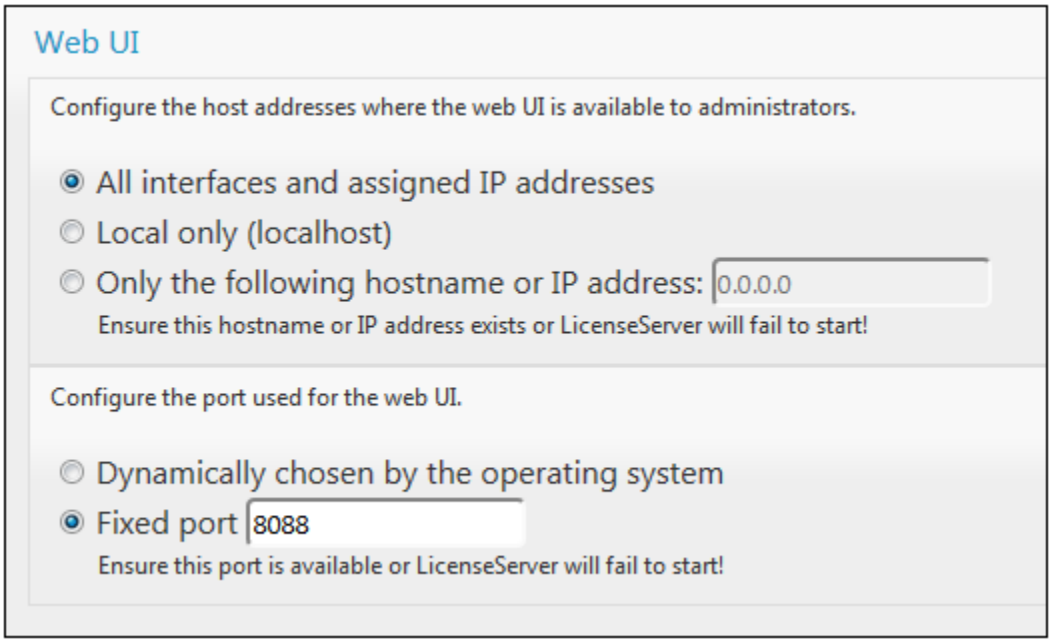
Nota: la página de configuración también se puede abrir directamente desde el icono **Finder | Aplicaciones | Altova License Server**.

Iniciar sesión con la contraseña inicial
Después de seguir todas estas instrucciones se abre la página de configuración por la página de inicio de sesión (*imagen siguiente*). Puede iniciar sesión con la contraseña predeterminada `default`. Después de iniciar sesión puede cambiar la contraseña en la pestaña **Settings**.

 © 2013 Altova GmbH



Establecer un puerto fijo o dinámico para la página de configuración
El puerto de la página de configuración (y, por consiguiente, su dirección) se puede especificar en la pestaña **Settings**. El puerto predeterminado es `8088` pero puede definir cualquier puerto para la página de configuración de LicenseServer (*imagen siguiente*). También puede permitir que el puerto se seleccione dinámicamente cada vez que se inicie LicenseServer. En este caso, necesitará buscar la URL de la página de configuración en el archivo `WebUI.html` (consulte los apartados Abrir la página de configuración de LicenseServer (Windows) y Abrir la página de configuración de LicenseServer (Linux)).



La ventaja de usar un puerto fijo es que la URL de la página se conoce por adelantado y se puede abrir con más facilidad. Si el puerto se asigna dinámicamente, la parte del puerto de la URL debe buscarse en el archivo `WebUI.html` cada vez que se inicia LicenseServer.

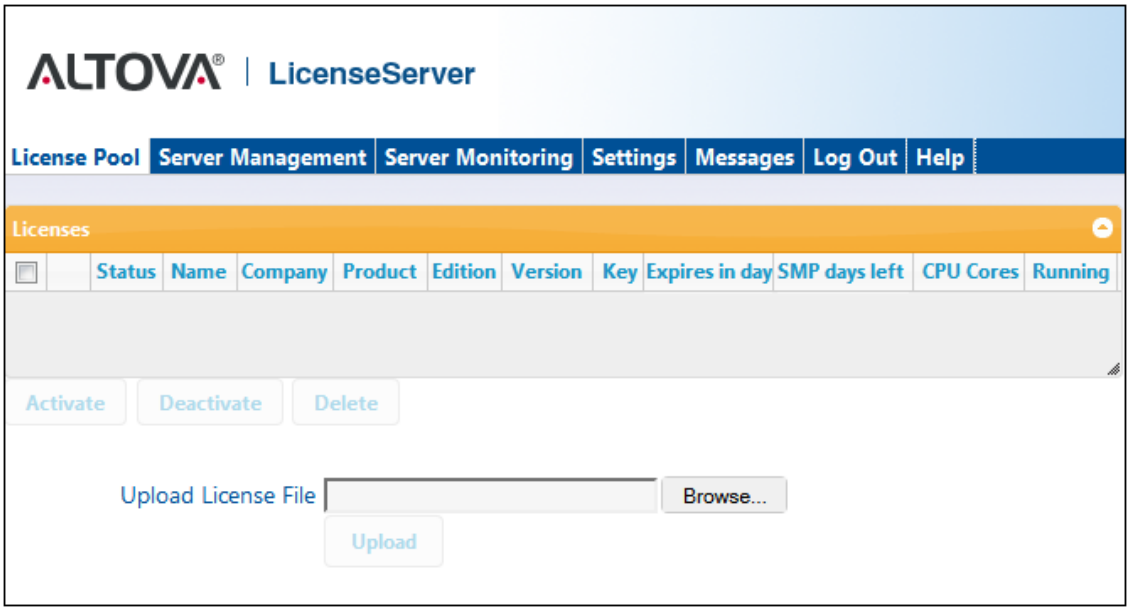
10.6.5 Cargar licencias en LicenseServer

Este apartado explica:

- Cómo cargar un archivo de licencias al repertorio de licencias de LicenseServer
- El estado de las licencias
- Cómo activar las licencias que desea usar
- Pasos siguientes

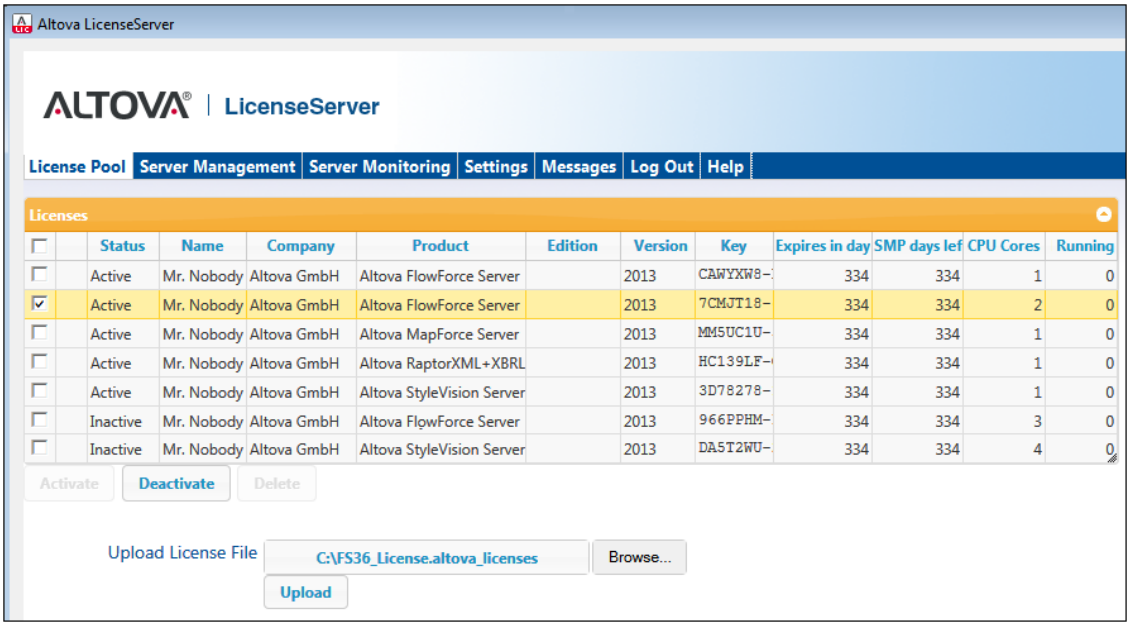
Cargar un archivo de licencias al repertorio de licencias de LicenseServer
Cuando reciba el archivo de licencias de Altova, cárguelo al repertorio de licencias de Altova LicenseServer (*ver instrucciones más abajo*). Un archivo de licencias puede contener varias licencias y al cargarlo, se cargan todas las licencias incluidas en el archivo. Después podrá asignar estas licencias a los productos de Altova que estén registrados con el servidor de LicenseServer. Las licencias se almacenan en LicenseServer en un repertorio de licencias. Para acceder al repertorio de licencias abra la pestaña **License Pool** de la página de configuración (*imagen siguiente*).

Los archivos de licencias se cargan a LicenseServer haciendo clic en el botón **Upload** (Cargar) de la pestaña **License Pool** (*imagen siguiente*).



Primero haga clic en el botón **Browse** para buscar el archivo de licencias que desea cargar. El archivo seleccionado aparece en el campo de entrada *Upload License File* y se habilita el botón **Upload**. Haga clic en el botón. Todas las licencias del archivo se cargan al repertorio y aparecen en la tabla de la pestaña **License Pool**, como en el ejemplo de la imagen siguiente.

© 2013 Altova GmbH



Estado de las licencias
Las licencias pueden tener varios estados:

- ***Activating*** *(En activación):* cuando se carga una licencia al repertorio de licencias de LicenseServer, el servidor transmite datos sobre la licencia al servidor maestro `altova.com` para validarla, verificarla y activarla. Este paso es necesario para comprobar que se cumple el contrato de licencia de Altova. Mientras se completa esta transacción (que puede durar entre 30 segundos y varios minutos, dependiendo de la velocidad y del tráfico de la conexión) el estado de la licencia es ***Activating...*** (*En activación...*).

- ***Failed verification*** *(No se puede verificar):* si no se puede establecer la conexión con el servidor maestro `altova.com`, el estado de la licencia es ***Failed verification*** *(No se puede verificar).* Si esto ocurre, revise su conexión a Internet y las reglas de su servidor de seguridad y compruebe que LicenseServer puede comunicarse con el servidor maestro `altova.com`.

- ***Active*** *(Activa):* la licencia se activa tras ser verificada.

- ***Inactive*** *(No activa):* si la licencia se verifica pero está presente en otro LicenseServer de la red, su estado es ***Inactive*** *(No activa)*. Esto también puede ocurrir cuando el administrador desactiva la licencia a mano en el repertorio de licencias.

- ***Blocked*** *(Bloqueada):* la licencia tiene este estado cuando se produce un error al verificar la licencia y el servidor maestro `altova.com` no concede permiso a LicenseServer para usar la licencia. Esto puede deberse a una infracción del contrato de licencia o a un uso excesivo de una licencia, por ejemplo. Si su licencia está bloqueada, póngase en contacto con el equipo de soporte técnico de Altova.

A continuación resumimos la información anterior:

Estado	Significado
Activating *(En activación).*	Al cargar un archivo de licencias, los datos se envían a `altova.com` para verificar las licencias. Actualice el explorador para ver el estado de las licencias. El proceso de verificación y activación puede durar unos minutos.
Failed verification *(No se puede verificar)*	No se puede establecer la conexión con `altova.com`. Cuando consiga establecer la conexión, vuelve a iniciar LicenseServer o intente activar la licencia otra vez (con el botón **Activate**).
Active *(Activa)*	La licencia se verificó correctamente y la licencia está activa.
Inactive *(No activa)*	La licencia se verificó correctamente pero la licencia está en otro servidor LicenseServer de la red. Las licencias se pueden desactivar con el botón **Deactivate**.
Blocked *(Bloqueada)*	No se puede verificar la licencia. La licencia no es válida y se bloqueó. Póngase en contacto con el equipo de soporte técnico de Altova.

Nota: tras enviar los datos de licencia a `altova.com` es necesario actualizar la ventana del explorador para actualizar el estado. La verificación y activación puede tardar unos minutos.

Nota: si no se puede establecer la conexión con `altova.com`, el estado de la licencia es ***Failed verification*** *(No se puede verificar)*. Cuando consiga establecer la conexión, vuelve a iniciar LicenseServer o intente activar la licencia otra vez (con el botón **Activar**).

Nota: cuando la licencia tiene el estado ***Inactive*** *(No activa)* o ***Blocked*** *(Bloqueada)* aparece un mensaje explicando el motivo en la pestaña **Messages**.

A los productos solamente se les pueden asignar licencias activas. Las licencias activas se pueden activar o eliminar del repertorio de licencias. Si una licencia se elimina del repertorio, se puede cargar otra vez cargando el archivo de licencias correspondiente. Cuando se actualiza un archivo de licencias, solamente se cargan al repertorio las licencias que todavía no están en él. Para activar, desactivar o eliminar una licencia, selecciónela y haga clic en el botón **Activate** , **Deactivate** o **Delete** respectivamente.

Activar las licencias que desea usar

Antes de asignar una licencia a un producto de Altova es necesario activarla. Si está inactiva, selecciónela y haga clic en el botón **Activate**.

¿Y ahora?

Tras cargar el archivo de licencias al repertorio de LicenseServer y comprobar que las licencias que quiere usar están activas:

1. Registre el producto servidor de Altova (FlowForce Server, MapForce Server, StyleVision Server) con LicenseServer. Si lo hizo antes de cargar las licencias, ya puede empezar a asignar las licencias.
2. Asigne una licencia al producto de Altova que está registrado con LicenseServer.

 © 2013 Altova GmbH

10.6.6 Registrar FlowForce Server con LicenseServer

Este apartado explica:

- Cómo registrar FlowForce Server con LicenseServer
- Cómo abrir la página de configuración de FlowForce Server (Windows)
- Cómo abrir la página de configuración de FlowForce Server (Linux)
- Cómo registrar FlowForce Server desde la página de configuración (Windows/Linux)
- Cómo registrar FlowForce Server desde la línea de comandos (Windows)
- Cómo registrar FlowForce Server desde la línea de comandos (Linux)
- Pasos siguientes

Cómo registrar FlowForce Server con LicenseServer
Puede registrar FlowForce Server con LicenseServer de dos maneras:

- Desde la página de configuración de FlowForce Server (Windows/Linux)
- Desde la interfaz de la línea de comandos de FlowForce Server (Windows)
- Desde la interfaz de la línea de comandos de FlowForce Server (Linux)

Abrir la página de configuración de FlowForce Server (Windows)
Puede abrir la página de configuración de FlowForce Server de dos maneras:

- Desde el menú **Inicio**:
  Inicio | **Altova FlowForce Server 2013 | Página de configuración de FlowForce Server**

- Con Altova ServiceController: haga clic en el icono de ServiceController de la bandeja del sistema. En el menú que aparece haga clic en **Altova FlowForce Web | Setup**. Aparece la página de configuración de FlowForce Server (*imagen anterior*).

Abrir la página de configuración de FlowForce Server (Linux)
Tras instalar FlowForce Server en Linux (consulte el manual de FlowForce Server para obtener más información), inicie FlowForce Web Server como servicio con este comando:

```
sudo /etc/init.d/flowforcewebserver start
```

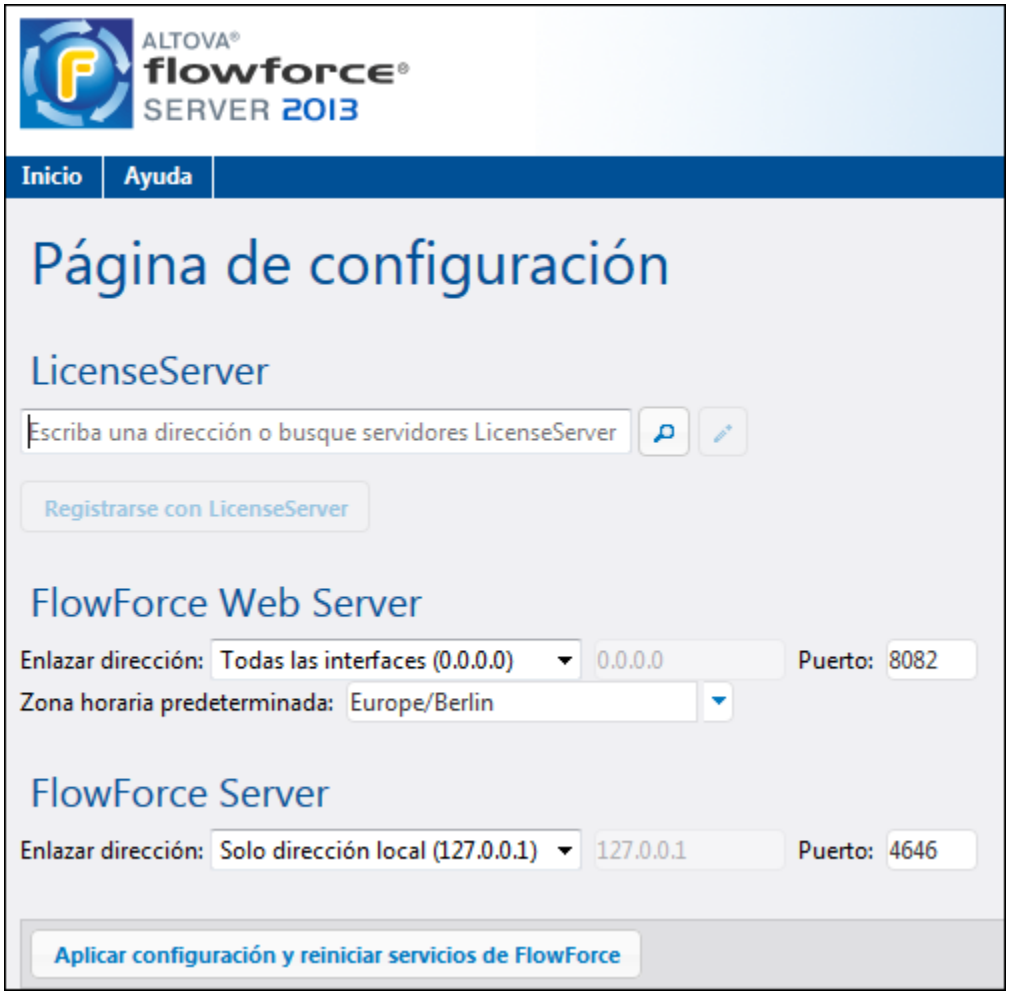
En la ventana de la Terminal aparece un mensaje con la URL en la que se está ejecutando FlowForceWeb:

FlowForceWeb running on `http://127.0.1.1:3459/setup?key=52239315203`

Copie la URL que aparece en el mensaje y péguela en la barra de dirección de su explorador web. Aparece la página de configuración de FlowForce Server (*imagen siguiente*).

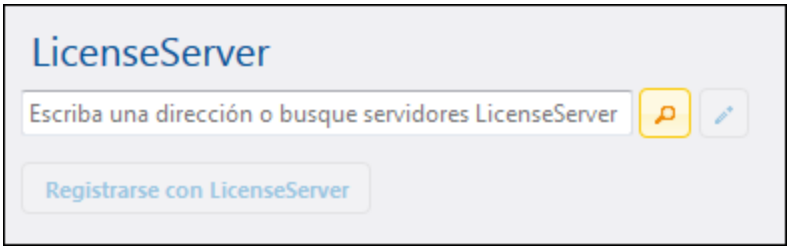
Registrar FlowForce Server desde la página de configuración (Windows/Linux)
En la página de configuración (*imagen siguiente*) en el campo *LicenseServer* se debe indicar el

servidor Altova LicenseServer con el que desea registrar el producto.



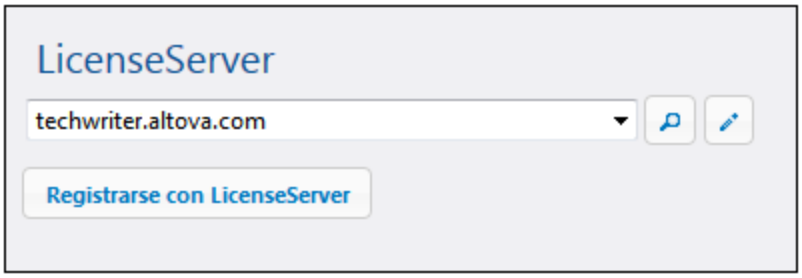
Puede rellenar este campo de dos maneras:

- Puede buscar servidores LicenseServer en la red (los que estén en ejecución). Para ello haga clic en el botón en forma de lupa (*resaltado en amarillo en la imagen siguiente*).

  La búsqueda devuelve una lista con los servidores LicenseServer que están ejecutándose en la red. Seleccione el LicenseServer que almacena su licencia de FlowForce Server (*imagen siguiente*).

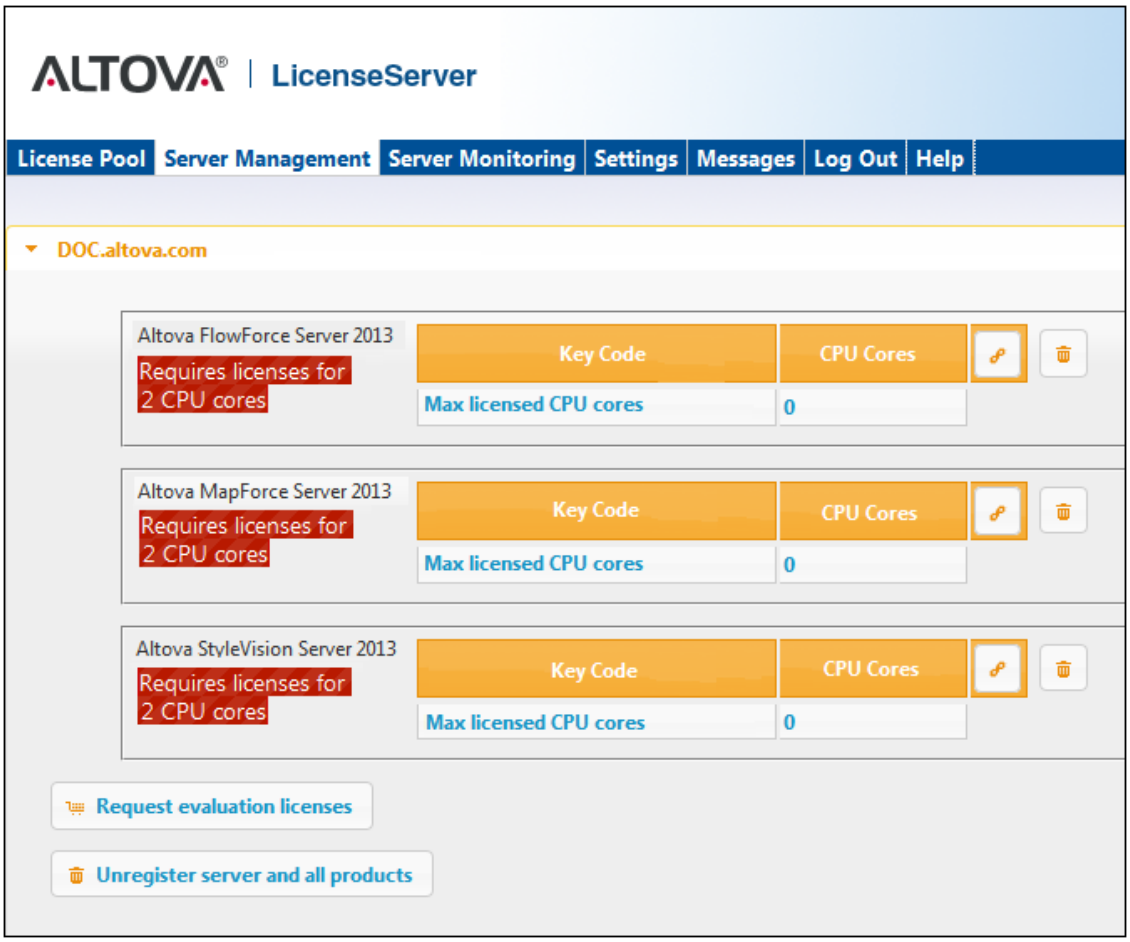
 © 2013 Altova GmbH



- O puede introducir la dirección del servidor LicenseServer directamente en el campo *LicenseServer*. Si está disponible una lista de servidores, haga clic en el botón en forma de lápiz para introducir la dirección en el campo *LicenseServer*.

Tras especificar el servidor LicenseServer, haga clic en el botón **Registrarse con LicenseServer**. FlowForce Server se registra con el LicenseServer indicado y se abre la página de configuración de LicenseServer por la pestaña de gestión de servidores **Server Management** (*imagen siguiente*).

Nota: quizá sea necesario permitir páginas emergentes en el explorador web.



En el ejemplo de la imagen anterior puede ver que se registraron tres productos con el servidor Altova LicenseServer `techwriter.altova.com`. El apartado Asignar licencias a productos

registrados explica cómo se asignan las licencias.

Registrar FlowForce Server desde su interfaz de la línea de comandos (Windows)
En los equipos Windows también puede registrar FlowForce Server con un LicenseServer de la red desde la línea de comandos usando el comando `licenseserver`:

```
FlowForceServer licenseserver Servidor-O-Dirección-IP
```

Por ejemplo, si LicenseServer se está ejecutando en `http://localhost:8088`, entonces debe registrar FlowForce Server con:

```
FlowForceServer licenseserver localhost
```

Si el paquete de instalación de FlowForce Server incluía otros productos servidor de Altova, al registrar FlowForce Server se registran automáticamente los demás productos. Tras registrar FlowForce Server, vaya a LicenseServer y asígnele una licencia. Para más información consulte el apartado Asignar licencias a productos registrados.

Registrar FlowForce Server desde su interfaz de la línea de comandos (Linux)
En los equipos Windows también puede registrar FlowForce Server con un LicenseServer desde la línea de comandos usando el comando `licenseserver`. Recuerde que FlowForce Server debe iniciarse con privilegios de root:

```
sudo /opt/Altova/FlowForceServer2013/bin/flowforceserver licenseserver
localhost
```

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

```
/opt/Altova/MapForceServer2013/bin
```

Cuando termine de registrar FlowForce Server, vaya a LicenseServer y asígnele una licencia. Para más información consulte el apartado Asignar licencias a productos registrados.

¿Y ahora?

1. Si todavía no ha cargado su archivo de licencias en LicenseServer (ver el apartado Cargar licencias en LicenseServer), cárguelo ahora y compruebe si la licencia que quiere usar está activa.
2. Asigne una licencia a su producto de Altova.

© 2013 Altova GmbH

10.6.7 Registrar MapForce Server con LicenseServer

Este apartado explica:

- Cómo registrar MapForce Server desde FlowForce Server (Windows)
- Cómo registrar MapForce Server (Windows)
- Cómo registrar MapForce Server (Linux)
- Pasos siguientes

MapForce Server se puede instalar como componente de FlowForce o como producto servidor independiente. En ambos casos es necesario registrarlo con Altova LicenseServer. Una vez registrado se le puede asignar una licencia desde LicenseServer. Si instala MapForce Server como componente de FlowForce (desde el mismo paquete de instalación) en un sistema Windows, MapForce Server se registra automáticamente al registrar FlowForce Server. En sistemas Linux, sin embargo, MapForce Server se registra automáticamente al registrar FlowForce Server solamente si instaló MapForce Server después de instalar FlowForce Server.

Registrar MapForce Server desde FlowForce Server (Windows)
MapForce Server es un componente de FlowForce Server así que, al registrar FlowForce Server con un LicenseServer de la red, MapForce Server se registra con el mismo LicenseServer automáticamente. Para más información consulte Registrar FlowForce Server con LicenseServer.

Una vez registrado, asígnele una licencia desde LicenseServer. Para más información consulte Asignar licencias a productos registrados.

Registrar MapForce Server como producto independiente (Windows)
Si instaló MapForce Server como producto independiente, regístrelo con un Altova LicenseServer de su red y después asígnele una licencia. Para registrar MapForce Server desde la línea de comandos use el comando `licenseserver`:

```
MapForceServer licenseserver Servidor-o-Dirección-IP
```

Por ejemplo, si LicenseServer se está ejecutando en `http://localhost:8088`, entonces debe registrar MapForce Server con:

```
MapForceServer licenseserver localhost
```

Una vez registrado, asígnele una licencia desde LicenseServer. Para más información consulte Asignar licencias a productos registrados.

Registrar MapForce Server (Linux)
En equipos Linux utilice el comando `licenseserver` para registrar MapForce Server con LicenseServer desde la línea de comandos. Recuerde que debe iniciar MapForce Server con privilegios de root:

```
sudo /opt/Altova/MapForceServer2013/bin/mapforceserver licenseserver
localhost
```

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

```
/opt/Altova/MapForceServer2013/bin
```

Una vez registrado, asígnele una licencia desde LicenseServer. Para más información consulte Asignar licencias a productos registrados.

¿Y ahora?

1. Si todavía no ha cargado su archivo de licencias en LicenseServer (ver el apartado Cargar licencias en LicenseServer), cárguelo ahora y compruebe si la licencia que quiere usar está activa.
2. Asigne una licencia a su producto de Altova que registró con el servidor LicenseServer.

© 2013 Altova GmbH

10.6.8 Registrar RaptorXML(+XBRL) Server con LicenseServer

Este apartado explica:

- Cómo registrar RaptorXML(+XBRL) Server (Windows)
- Cómo registrar RaptorXML(+XBRL) Server (Linux)
- Pasos siguientes

Primero debe instalar RaptorXML(+XBRL) Server en el equipo servidor e iniciarlo como servicio. Después debe registrarlo con LicenseServer. Una vez registrado, podrá asignarle una licencia desde LicenseServer. Esta sección explica cómo registrar RaptorXML(+XBRL) Server con LicenseServer.

Registrar RaptorXML(+XBRL) Server (Windows)

Puede registrar RaptorXML(+XBRL) Server desde la interfaz de la línea de comandos con ayuda del comando `licenseserver`:

```
RaptorXML licenseserver Servidor-O-Dirección-IP
```

Por ejemplo, si LicenseServer se está ejecutando en `http://localhost:8088`, entonces debe registrar RaptorXML(+XBRL) Server con:

```
RaptorXML licenseserver localhost
```

Tras registrar RaptorXML(+XBRL) Server, abra la interfaz de administración de LicenseServer y asigne una licencia a RaptorXML(+XBRL) Server. Esto se describe en el apartado Asignar licencias a productos registrados.

Registrar RaptorXML(+XBRL) Server (Linux)

En los equipos Linux puede registrar RaptorXML(+XBRL) Server con LicenseServer con ayuda del comando `licenseserver` de la interfaz de la línea de comandos de RaptorXML(+XBRL) Server. Recuerde que para ello es necesario iniciar RaptorXML(+XBRL) Server con privilegios root.

```
sudo /opt/Altova/RaptorXMLServer2013/bin/raptorxmlserver licenseserver
localhost
```

En el comando anterior **`localhost`** es el nombre del servidor en el que está instalado LicenseServer. Además observe que la ubicación del ejecutable de RaptorXML(+XBRL) Server es:

`/opt/Altova/``RaptorXMLServer`**`2013/bin`**

Tras registrar RaptorXML(+XBRL) Server, abra la interfaz de administración de LicenseServer y asigne una licencia a RaptorXML(+XBRL) Server. Esto se describe en el apartado Asignar licencias a productos registrados.

¿Y ahora?

1. Si todavía no ha cargado su archivo de licencias en LicenseServer (ver el apartado Cargar licencias en LicenseServer), cárguelo ahora y compruebe si la licencia que quiere usar está activa.
2. Asigne una licencia a su producto de Altova que registró con el servidor LicenseServer.

© 2013 Altova GmbH

10.6.9 Registrar StyleVision Server con LicenseServer

Este apartado explica:

- Cómo registrar StyleVision Server desde FlowForce Server (Windows)
- Cómo registrar StyleVision Server (Windows)
- Cómo registrar StyleVision Server (Linux)
- Pasos siguientes

StyleVision Server se puede instalar como componente de FlowForce o como producto servidor independiente. En ambos casos es necesario registrarlo con Altova LicenseServer. Una vez registrado se le puede asignar una licencia desde LicenseServer. Si instala StyleVision Server como componente de FlowForce (desde el mismo paquete de instalación) en un sistema Windows, StyleVision Server se registra automáticamente al registrar FlowForce Server. En sistemas Linux, sin embargo, StyleVision Server se registra automáticamente al registrar FlowForce Server solamente si instaló StyleVision Server después de instalar FlowForce Server.

Registrar StyleVision Server desde FlowForce Server (Windows)
StyleVision Server es un componente de FlowForce Server así que, al registrar FlowForce Server con un LicenseServer de la red, StyleVision Server se registra automáticamente con el mismo LicenseServer. Para más información consulte Registrar FlowForce Server con LicenseServer.

Una vez registrado, asígnele una licencia desde LicenseServer. Para más información consulte Asignar licencias a productos registrados.

Registrar StyleVision Server como producto independiente (Windows)
Si instaló StyleVision Server como producto independiente, regístrelo con un Altova LicenseServer de su red y después asígnele una licencia. Para registrar StyleVision Server desde la línea de comandos use el comando `licenseserver`:

```
StyleVisionServer licenseserver Servidor-o-Dirección-IP
```

Por ejemplo, si LicenseServer se está ejecutando en `http://localhost:8088`, entonces debe registrar StyleVision Server con:

```
StyleVisionServer licenseserver localhost
```

Una vez registrado, asígnele una licencia desde LicenseServer. Para más información consulte Asignar licencias a productos registrados.

Registrar StyleVision Server (Linux)
En equipos Linux utilice el comando `licenseserver` para registrar StyleVision Server con LicenseServer desde la línea de comandos. Recuerde que debe iniciar StyleVision Server con privilegios de root:

```
sudo /opt/Altova/StyleVisionServer2013/bin/stylevisionserver licenseserver
localhost
```

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

```
/opt/Altova/StyleVisionServer2013/bin
```

Una vez registrado, asígnele una licencia desde LicenseServer. Para más información consulte Asignar licencias a productos registrados.

¿Y ahora?

1. Si todavía no ha cargado su archivo de licencias en LicenseServer (ver el apartado Cargar licencias en LicenseServer), cárguelo ahora y compruebe si la licencia que quiere usar está activa.
2. Asigne una licencia a su producto de Altova.

© 2013 Altova GmbH

10.6.10 Asignar licencias a productos registrados

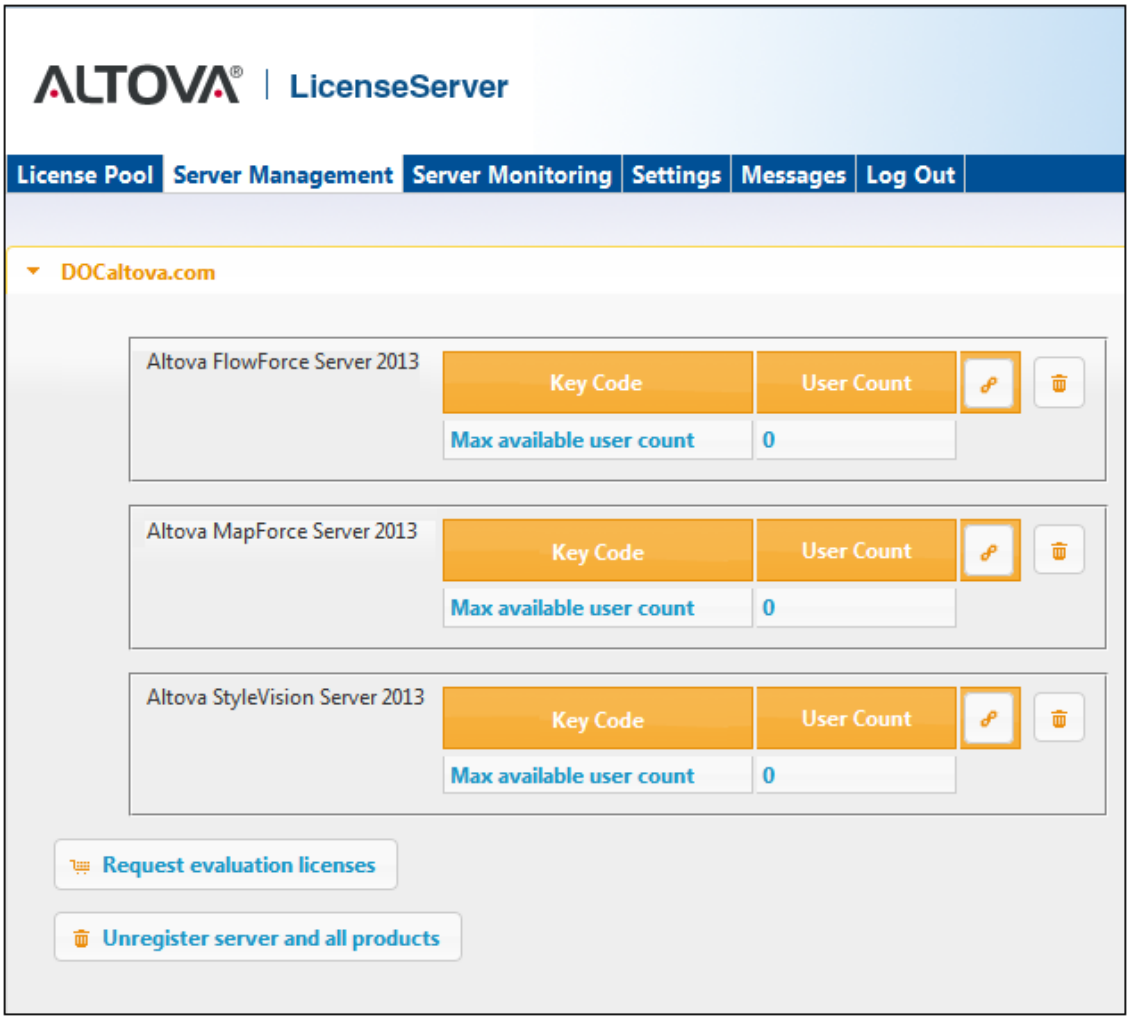
Este apartado explica:

- Los pasos previos a la asignación de licencias
- La pestaña **Server Management**
- Los iconos de la pestaña **Server Management**
- Núcleos y licencias
- Cómo asignar una licencia
- Cómo quitar productos del registro de LicenseServer

Pasos previos a la asignación de licencias
Antes de asignar una licencia a su producto de Altova, compruebe que:

- La licencia pertinente está cargada en el repertorio de licencias de LicenseServer y que la licencia está activa.
- Su producto de Altova está registrado con LicenseServer.

Pestaña Server Management
Las licencias se asignan en la pestaña de gestión de servidores **Server Management** de la página de configuración de LicenseServer (*imagen siguiente*). En el ejemplo de la imagen se registraron tres productos de Altova con LicenseServer. Puesto que el paquete de instalación de FlowForce Server incluye MapForce Server y StyleVision Server, al registrar FlowForce Server con LicenseServer, en el mismo LicenseServer se registran automáticamente MapForce Server y StyleVision Server. Es decir, si registra FlowForce Server no hace falta registrar los otros dos productos por separado.



Es necesario mencionar estos aspectos de la pestaña **Server Management**:

- cada producto aparece bajo el nombre de su equipo cliente. En el ejemplo de la imagen, aparece un equipo cliente llamado `DOCaltova.com`. Este equipo cliente (`DOCaltova.com`) tiene registrados tres productos de Altova con el servidor LicenseServer. Si otro equipo cliente tiene un producto registrado con este LicenseServer, el equipo cliente y sus productos registrados aparecerán en la pestaña **Server Management**.
- junto a los productos de Altova del equipo cliente aparece la entrada *Key code*, que toma el código clave de una licencia. El código clave del producto registrado se asigna haciendo clic en el botón en forma de nudo (**Editar licencias asignadas**) y seleccionando la licencia necesaria del repertorio de licencias. Este procedimiento se explica más detenidamente un poco más adelante.
- junto a cada producto aparece el número de núcleos que debe tener licencia para poder ejecutar el producto en el equipo cliente. Si el número de núcleos de la licencia es inferior al número de núcleos del equipo, los datos aparecen en rojo (*imagen anterior*). El número de núcleos que necesitan una licencia es el número de núcleos del cliente y LicenseServer obtiene la información del equipo cliente.

Iconos de la pestaña Server Management

Editar licencias asignadas: abre el cuadro de diálogo de gestión de licencias, donde puede asignar nuevas licencias al producto y editar las licencias asignadas previamente.

Mostrar licencias: aparece con cada licencia. Abre la pestaña del repertorio de licencias y resalta la licencia seleccionada para que pueda ver sus detalles.

Quitar producto del registro: elimina el producto seleccionado (del equipo cliente seleccionado) del registro de LicenseServer.

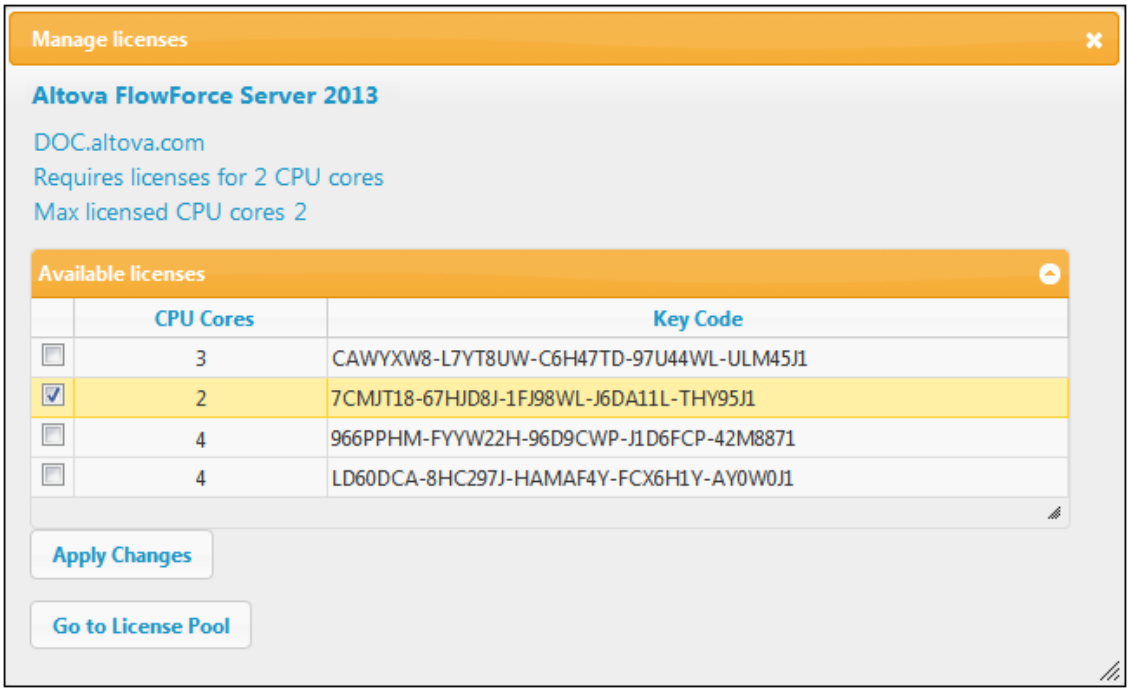
Nota sobre núcleos y licencias
La asignación de licencias a productos servidor de Altova depende de cuántos núcleos de procesador tiene el equipo donde se ejecuta el producto servidor de Altova. Por ejemplo, un procesador dual tiene dos núcleos, un procesador *quad* tiene cuatro núcleos, un procesador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

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 será menos rápida que si utilizara todos los núcleos disponibles en el servidor.

> **Nota:** cada licencia se puede usar solamente para un equipo cliente, incluso si no se usa la totalidad de la licencia. 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 para otro equipo cliente simultáneamente.

Asignar una licencia
Para asignar una licencia a un producto registrado, haga clic en el botón **Editar licencias asignadas** del producto. Aparece el cuadro de diálogo de gestión de licencias "Manage Licenses" (*imagen siguiente*).



En este cuadro de diálogo:

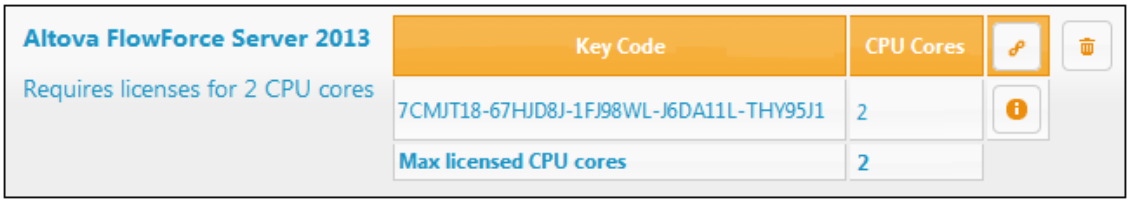
- El producto al que se debe asignar la licencia aparece en negrita en la parte superior izquierda del cuadro de diálogo. Por ejemplo, en la imagen anterior se trata de **Altova FlowForce Server 2013**.
- Las licencias que están activas en el repertorio para este producto aparecen enumeradas en la tabla "Available licenses". Por ejemplo, en la imagen anterior hay cuatro licencias activas para FlowForce Server en el repertorio de licencias. LicenseServer detecta automáticamente para qué producto se emitió cada licencia del repertorio.
- Las licencias que aparecen en la imagen anterior son para procesadores de 3, 2, 4 y 4 núcleos respectivamente.
- Debe conocer el número de núcleos que tiene el servidor en el que está instalado el producto servidor de Altova. Si el equipo tiene un procesador dual, necesitará una licencia para dos núcleos. Para ello podría usar la segunda licencia que aparece seleccionada en la imagen. Pero también podría combinar varias licencias. Por ejemplo, si el procesador es octa-core (8 núcleos) puede combinar dos licencias para 4 núcleos, como la tercera y cuarta licencia de la imagen.
- El diálogo "Manage Licenses" solamente enumera las licencias que están activas para este producto. No aparecen licencias de otros productos.
- La casilla de las licencias ya asignadas previamente (por ejemplo en otra instalación del mismo producto en la red) aparecen marcadas. Es decir, solamente puede seleccionar las licencias cuya casilla no esté marcada.
- La columna *CPU Cores* indica para cuántos núcleos es la licencia.
- Si desea realizar cambios en el repertorio de licencias (para cargar, activar, desactivar o eliminar licencias) haga clic en el botón **Go to License Pool** para ir al repertorio de licencias.

Seleccione la licencia que desea asignar. Observe que la casilla de la licencia seleccionada ahora aparece marcada y que el número máximo de núcleos para esa licencia aparece en la parte superior izquierda del diálogo (*Max licensed CPU Cores)*. Puede seleccionar más licencias si quiere incrementar el número de núcleos para el producto y el cliente

 © 2013 Altova GmbH

seleccionados. El número máximo de núcleos es la suma de los núcleos de todas las licencias seleccionadas.

Tras seleccionar las licencias haga clic en **Apply changes** para aplicar los cambios realizados en el diálogo. Las licencias se asignan al producto y aparecen en la tabla de la pestaña **Server Management** (*imagen siguiente*). En el ejemplo de la imagen se asignó una licencia de dos núcleos a Altova FlowForce Server (en el equipo cliente `Doc.altova.com`).



Quitar productos del registro
Los productos de Altova registrados con LicenseServer se enumeran en la pestaña **Server Management** bajo el nombre del equipo cliente donde están instalados. Junto a ellos aparece el icono **Quitar del registro**. Haga clic en este icono para quitar el producto del registro. Si el producto tiene asignada una licencia, la asignación expira nada más quitar el producto del registro. Para quitar todos los productos del registro, haga clic en el botón **Unregister server and all products**, situado al final de la página (*primera imagen de este apartado*).

Para volver a registrar el producto, abra su página de configuración.

10.7 Referencia de la página de configuración

La página de configuración de LicenseServer es la interfaz del administrador con el servidor de licencias. Permite gestionar el servidor y asignar licencias a los productos de Altova que se registraron con LicenseServer (FlowForce Server, MapForce Server y StyleVision Server).

La página de configuración de LicenseServer se abre con un explorador web. Esto se explica en los apartados Abrir la página de configuración de LicenseServer (Windows) y Abrir la página de configuración de LicenseServer (Linux).

Esta sección es una referencia del usuario para la página de configuración y sus apartados se corresponden con las pestañas de la página de configuración:

- **Repertorio de licencias:** describe el uso pestaña **License Pool**.
- **Gestión de servidores:** describe el uso de la pestaña **Server Management**.
- **Supervisión de servidores:** describe el uso de la pestaña **Server Monitoring**.
- **Opciones de configuración:** describe el uso de la pestaña **Settings**.
- **Mensajes, Cerrar sesión**

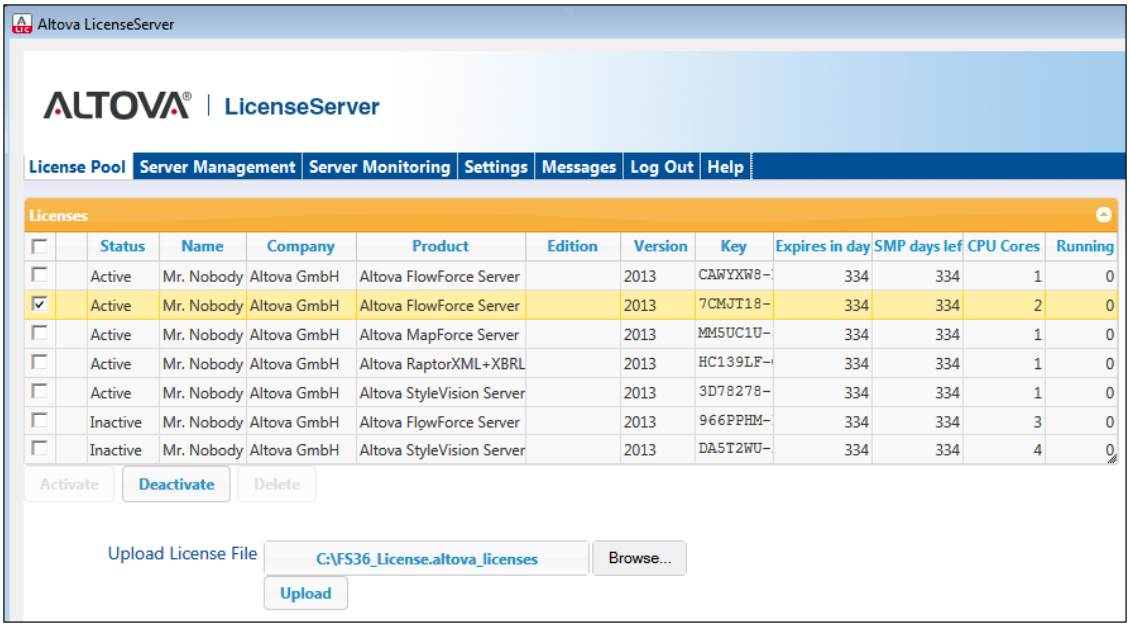
Consulte el apartado Asignar licencias para obtener más información sobre el proceso de asignación de licencias.

© 2013 Altova GmbH

10.7.1 Repertorio de licencias

La pestaña **License Pool** muestra todas las licencias que están en el servidor LicenseServer (*imagen siguiente*). Cuando se carga un archivo de licencias a LicenseServer con el botón **Cargar**, todas las licencias del archivo se colocan en el repertorio del servidor y aparecen en esta pestaña.

La pestaña del repertorio de licencias incluye información sobre todas las licencias que están en el servidor de licencias y, por tanto, ofrece un práctico resumen de todas las licencias disponibles. En esta pestaña también puede activar, desactivar y eliminar licencias.



Cargar licencias
Siga estos pasos para cargar un archivo de licencias (el que Altova GmbH le envía para su producto de Altova):

1. Haga clic en el botón **Browse** para buscar el archivo de licencias. Seleccione el archivo.
2. Haga clic en el botón **Upload** para cargar el archivo de licencias.
   Las licencias del archivo se colocan en el repertorio de licencias y aparecen en la tabla de la pestaña (*imagen anterior*).

Estado de las licencias
Las licencias pueden tener varios estados:

- ***Activating*** *(En activación):* cuando se carga una licencia al repertorio de licencias de LicenseServer, el servidor transmite datos sobre la licencia al servidor maestro `altova.com` para validarla, verificarla y activarla. Este paso es necesario para comprobar que se cumple el contrato de licencia de Altova. Mientras se completa esta transacción (que puede durar entre 30 segundos y varios minutos, dependiendo de la velocidad y del tráfico de la conexión) el estado de la licencia es ***Activating...*** (*En activación...*).

- ***Failed verification*** *(No se puede verificar):* si no se puede establecer la conexión con el servidor maestro `altova.com`, el estado de la licencia es ***Failed verification*** *(No se puede verificar).* Si esto ocurre, revise su conexión a Internet y las reglas de su servidor de seguridad y compruebe que LicenseServer puede comunicarse con el servidor maestro `altova.com`.

- ***Active*** *(Activa):* la licencia se activa tras ser verificada.

- ***Inactive*** *(No activa):* si la licencia se verifica pero está presente en otro LicenseServer de la red, su estado es ***Inactive*** *(No activa)*. Esto también puede ocurrir cuando el administrador desactiva la licencia a mano en el repertorio de licencias.

- ***Blocked*** *(Bloqueada):* la licencia tiene este estado cuando se produce un error al verificar la licencia y el servidor maestro `altova.com` no concede permiso a LicenseServer para usar la licencia. Esto puede deberse a una infracción del contrato de licencia o a un uso excesivo de una licencia, por ejemplo. Si su licencia está bloqueada, póngase en contacto con el equipo de soporte técnico de Altova.

A continuación resumimos la información anterior:

Estado	Significado
Activating *(En activación).*	Al cargar un archivo de licencias, los datos se envían a `altova.com` para verificar las licencias. Actualice el explorador para ver el estado de las licencias. El proceso de verificación y activación puede durar unos minutos.
Failed verification *(No se puede verificar)*	No se puede establecer la conexión con `altova.com`. Cuando consiga establecer la conexión, vuelve a iniciar LicenseServer o intente activar la licencia otra vez (con el botón **Activate**).
Active *(Activa)*	La licencia se verificó correctamente y la licencia está activa.
Inactive *(No activa)*	La licencia se verificó correctamente pero la licencia está en otro servidor LicenseServer de la red. Las licencias se pueden desactivar con el botón **Deactivate**.
Blocked *(Bloqueada)*	No se puede verificar la licencia. La licencia no es válida y se bloqueó. Póngase en contacto con el equipo de soporte técnico de Altova.

Nota: tras enviar los datos de licencia a `altova.com` es necesario actualizar la ventana del explorador para actualizar el estado. La verificación y activación puede tardar unos minutos.

Nota: si no se puede establecer la conexión con `altova.com`, el estado de la licencia es ***Failed verification*** *(No se puede verificar)*. Cuando consiga establecer la conexión, vuelve a iniciar LicenseServer o intente activar la licencia otra vez (con el botón **Activar**).

Nota: cuando la licencia tiene el estado ***Inactive*** *(No activa)* o ***Blocked*** *(Bloqueada)* aparece un mensaje explicando el motivo en la pestaña **Messages**.

A los productos solamente se les pueden asignar licencias activas. Las licencias activas se pueden activar o eliminar del repertorio de licencias. Si una licencia se elimina del repertorio, se puede cargar otra vez cargando el archivo de licencias correspondiente. Cuando se actualiza un archivo de licencias, solamente se cargan al repertorio las licencias que todavía no están en

él. Para activar, desactivar o eliminar una licencia, selecciónela y haga clic en el botón **Activate**, **Deactivate** o **Delete** respectivamente.

Conexión con el servidor maestro de licencias altova.com
Altova LicenseServer necesita comunicarse con el servidor maestro de licencias `altova.com` para validar y verificar los datos de licencia y para asegurarse de que los contratos de licencia de Altova se cumplen en todo momento. Esta comunicación se establece por HTTPS a través del puerto 443. Si tras completar la verificación inicial con el servidor maestro `altova.com`, Altova LicenseServer no puede volver a conectarse con `altova.com` durante más de 5 días (120 horas), se bloqueará el uso de los productos de Altova que estén conectados con Altova LicenseServer.

Cuando se pierde la conexión con el servidor maestro `altova.com`, la incidencia se registra en la pestaña **Messages** de la página de configuración de Altova LicenseServer. Además el administrador puede configurar Altova LicenseServer para que le envíe correos de alerta cuando se pierda la conexión con `altova.com`. Las opciones de configuración de estos correos de alerta se pueden definir en la pestaña **Settings** de la página de configuración.

Activar, desactivar y eliminar licencias
Para desactivar una licencia activa, selecciónela y haga clic en el botón **Deactivate**. Para activar una licencia inactiva, selecciónela y haga clic en el botón **Activate**. Para eliminarla haga clic en el botón **Delete**. Cuando se elimina una licencia, se quita del repertorio. Las licencias eliminadas se pueden añadir otra vez al repertorio cargando el archivo de licencias pertinente. Si vuelve a cargar un archivo de licencias, solamente se añaden las licencias que todavía no están en el repertorio. Las licencias que ya están no se vuelve a añadir.

Información sobre las licencias
La tabla donde se enumeran las licencias incluye estos datos:

- ***Status.*** Una licencia puede tener cinco estados distintos: *Failed verification* | *Activating* | *Active* | *Inactive* | *Blocked*. (*ver más arriba*).
- ***Name, Company***. Nombre y compañía del titular de la licencia. Estos datos se aportaron en el momento de compra.
- ***Product, Edition, Version***. Versión y edición de los productos con licencia.
- ***Key, Expires in days, SMP days left***. Clave de licencia que desbloquea el producto y el número de días que faltan para que expire la licencia. Cada producto con licencia incluye un paquete de soporte y mantenimiento, válido durante un plazo de tiempo. La columna *SMP* indica cuántos días faltan hasta la expiración del paquete de soporte y mantenimiento.
- ***CPU cores***. Número de núcleos de CPU permitidos por la licencia.
- ***Running***. Número de núcleos de CPU en ejecución que están utilizando la licencia en ese momento.

Nota sobre núcleos y licencias
La asignación de licencias a productos servidor de Altova depende de cuántos núcleos de procesador tiene el 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 licencia asignada a un producto instalado en un equipo servidor debe ser mayor o igual al número de núcleos disponibles en dicho servidor, ya sea un equipo virtual o uno físico.

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.

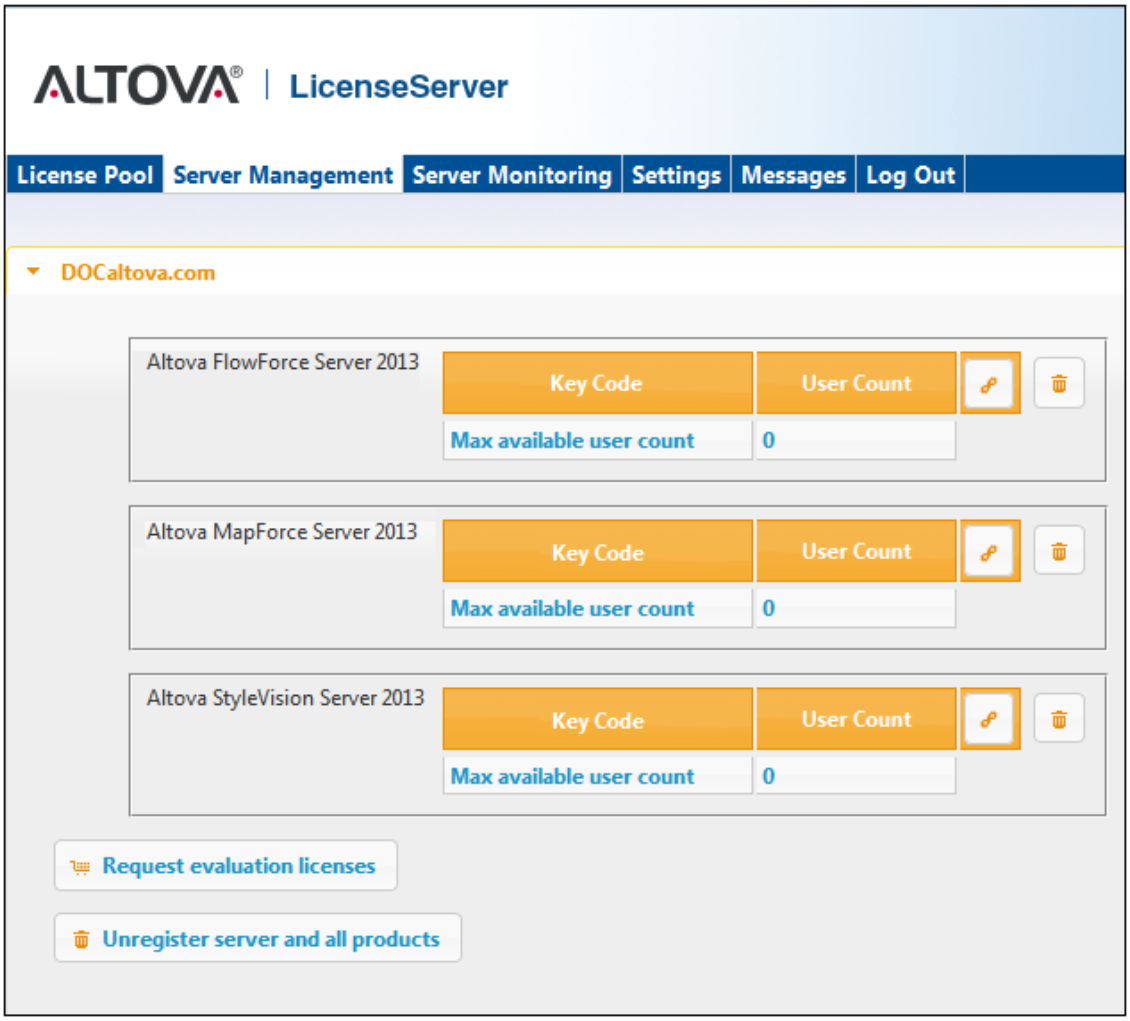
Si usa un equipo servidor con gran cantidad de núcleos, pero tiene un bajo volumen de procesamiento, también puede crear un equipo virtual que tenga adjudicados menos núcleos y comprar una licencia para ese menor número de núcleos. No obstante, dicha implementación será menos rápida que si utilizara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

> **Nota:** cada licencia se puede usar solamente para un equipo cliente, incluso si no se usa la totalidad de la licencia. 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 para otro equipo cliente simultáneamente.

 © 2013 Altova GmbH

10.7.2 Gestión de servidores

En la pestaña **Server Management** (*imagen siguiente*), puede asignar licencias a los productos registrados con el servidor de licencias.



En esta pestaña:

- cada producto aparece bajo el nombre de su equipo cliente. En el ejemplo de la imagen, aparece un equipo cliente llamado `DOCaltova.com`. Este equipo cliente (`DOCaltova.com`) tiene registrados tres productos de Altova con el servidor LicenseServer. Si otro equipo cliente tiene productos registrados con este mismo servidor de licencias, el equipo y sus productos registrados aparecerán en la pestaña **Server Management**.
- junto a los productos de Altova registrados del equipo cliente aparece la entrada *Key Code*, que toma el código clave de una licencia. El código clave del producto registrado se asigna haciendo clic en el botón en forma de nudo (**Editar licencias asignadas**) y seleccionando la licencia necesaria del repertorio de licencias. Este procedimiento se explica más detenidamente un poco más adelante.
- junto a cada producto aparece el número de núcleos que deben tener licencia para poder ejecutar le producto en el equipo cliente. Si el número de núcleos de la licencia es inferior al número de núcleos del equipo, los datos aparecen en rojo (*imagen anterior*). El número de núcleos que necesitan una licencia es el número de núcleos del

cliente y LicenseServer obtiene la información del equipo cliente.

Iconos de la pestaña Server Management

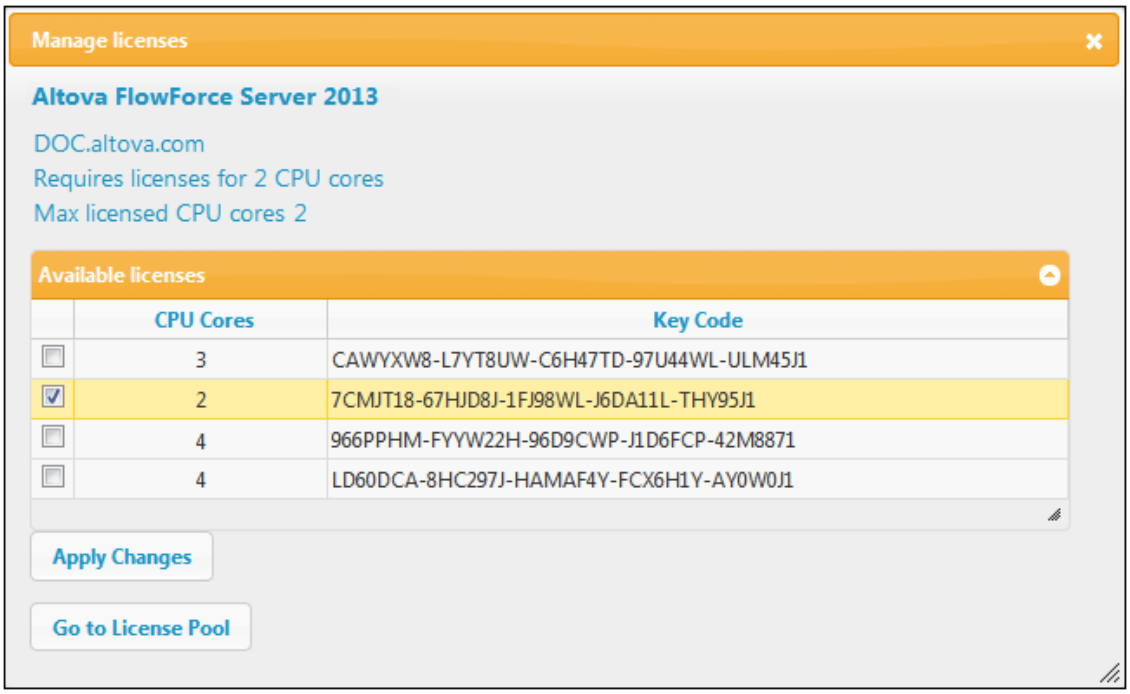
Editar licencias asignadas: abre el cuadro de diálogo de gestión de licencias, donde puede asignar nuevas licencias al producto y editar las licencias asignadas previamente.

Mostrar licencias: aparece con cada licencia. Abre la pestaña del repertorio de licencias y resalta la licencia seleccionada para que pueda ver sus detalles.

Quitar producto del registro: elimina el producto seleccionado (del equipo cliente seleccionado) del registro de LicenseServer.

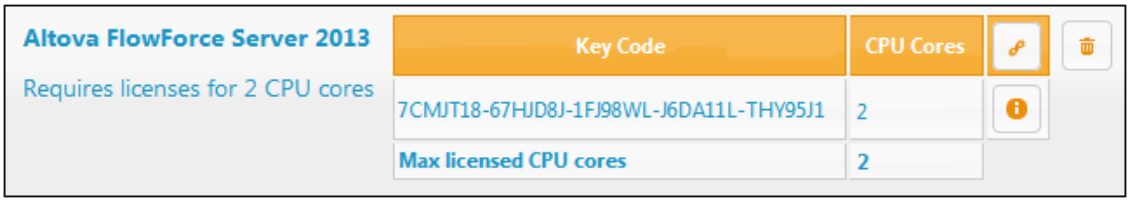
Asignar una licencia

Para asignar una licencia a un producto registrado, haga clic en el botón **Editar licencias asignadas** del producto. Aparece el cuadro de diálogo de gestión de licencias "Manage Licenses" (*imagen siguiente*).



Seleccione la licencia que desea asignar. Después haga clic en **Apply changes** para aplicar los cambios realizados. Las licencias se asignan al producto y aparecen en la tabla de la pestaña **Server Management** (*imagen siguiente*).

© 2013 Altova GmbH



Solicitar una licencia de evaluación
Altova ofrece licencias de evaluación gratuitas y válidas durante 30 días para que pueda probar todos los productos servidor. Primero debe descargar e instalar el producto servidor de Altova que desea probar y registrarlo con LicenseServer. Al final de la pestaña **Server Management**, haga clic en el botón **Request evaluation licenses** para solicitar licencias de evaluación. Se abre un cuadro de diálogo que enumera los productos servidor de Altova (del equipo cliente) que están registrados con LicenseServer. Marque las casillas de los productos que desea probar, rellene los campos de registro y envíe la solicitud. Poco después recibirá un correo electrónico con las licencias de evaluación, gratuitas y válidas durante 30 días. El número de núcleos para los que es válida la licencia de cada producto coincidirá con el número de núcleos que necesita el producto en el momento de realizarse la solicitud. Guarde las licencias en el disco y cárguelas al repertorio de licencias.

Quitar productos del registro
Los productos de Altova registrados con LicenseServer se enumeran en la pestaña **Server Management** bajo el nombre del equipo cliente donde están instalados. Junto a ellos aparece el icono **Quitar del registro**. Haga clic en este icono para quitar el producto del registro. Si el producto tiene asignada una licencia, la asignación expira al quitar el producto del registro. Para quitar todos los productos del registro, haga clic en el botón **Unregister server and all products** situado al final de la página (*primera imagen de este apartado*).

Para volver a registrar el producto, abra su página de configuración.

Para más información lea el apartado Asignar licencias a productos registrados.

10.7.3 Supervisión de servidores

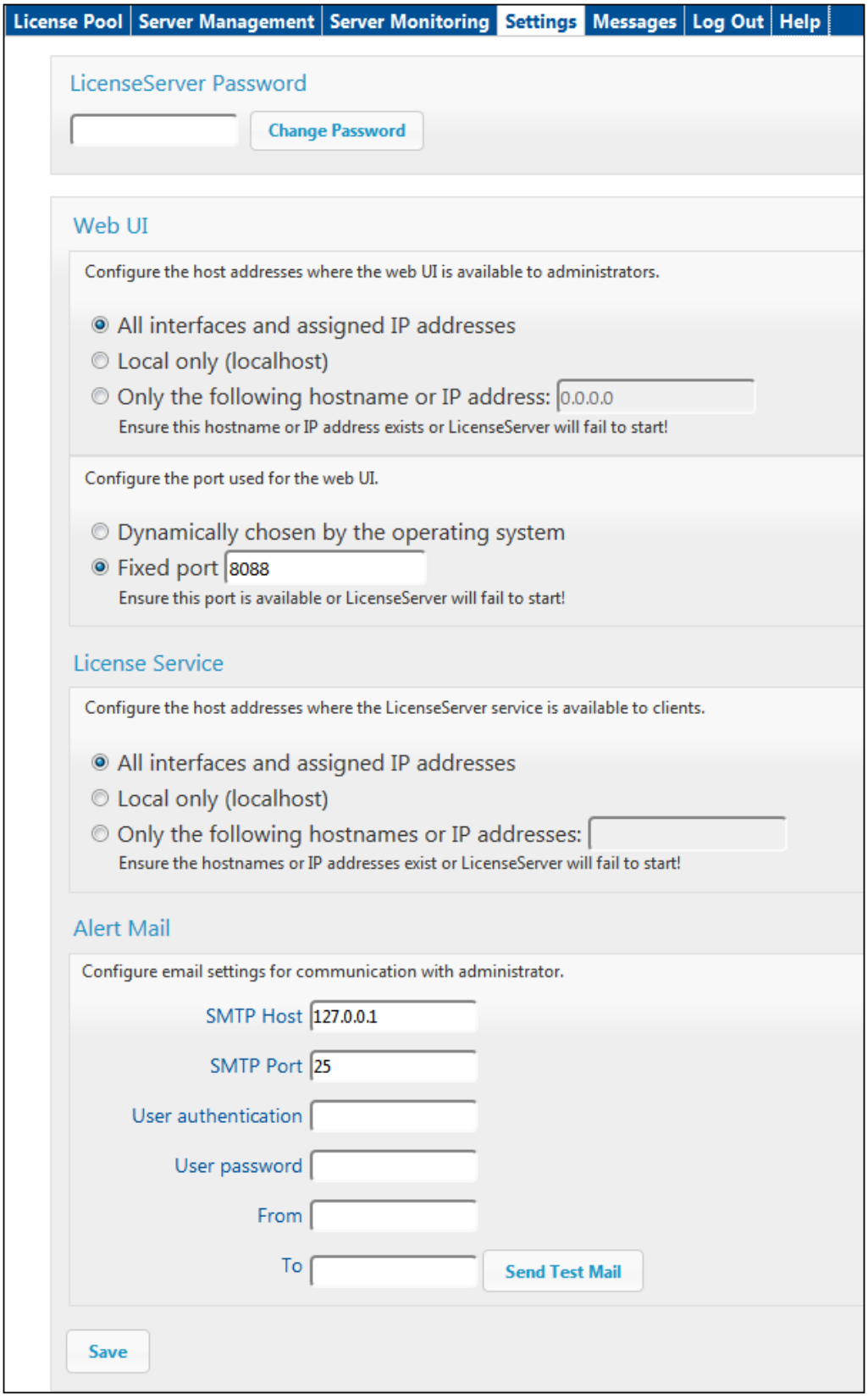
La pestaña **Supervisión de servidores** ofrece una visión de conjunto de los servidores que están ejecutando productos de Altova. Es decir, aquí encontrará información sobre cada producto, sobre los usuarios y las licencias.

© 2013 Altova GmbH

10.7.4 Opciones de configuración

En la pestaña **Settings** (*imagen siguiente*) puede definir las opciones de configuración del servidor LicenseServer:

- puede definir la contraseña necesaria para iniciar sesión en LicenseServer.
- puede definir opciones de red para la página de configuración web y para el servidor LicenseServer (*ver más abajo*).
- puede definir opciones para los correos de alerta (*ver más abajo*).



 © 2013 Altova GmbH

Configuración de red
El administrador puede especificar los puntos de acceso de red para la página de configuración web y para el servidor LicenseServer:

- *Web UI (interfaz web):* las direcciones IP permitidas pueden ser direcciones IP e interfaces del equipo o una dirección fija. Los puertos se pueden calcular de forma dinámica o fija. Esto permite un amplio abanico de opciones de configuración dirección IP:puerto. El puerto predeterminado es **8088**.
- *License Service (Servicio de licencias):* las direcciones IP pueden ser direcciones IP e interfaces del equipo o una dirección fija. El número de puerto es fijo: el puerto **35355**.

Estas opciones permiten un acceso no restringido a LicenseServer y a su página de configuración de forma predeterminada desde las redes a las que está conectado LicenseServer. Si quiere restringir el acceso a LicenseServer o a su página de configuración, defina las opciones correspondientes en esta pestaña y haga clic en **Save** para guardar.

Correos de alerta
Altova LicenseServer necesita estar conectado al servido maestro `altova.com`. Si la conexión se interrumpe más de 24 x 5 horas (es decir, 5 días), LicenseServer no permitirá licencias. Como resultado, puede que las sesiones de trabajo con productos Altova registrados con LicenseServer se interrumpan. Si quiere puede configurar LicenseServer para que envíe un correo de alerta al administrador cuando la conexión se interrumpa y en este grupo de opciones puede configurar los correos de alerta y la dirección de correo del administrador.

Los campos *SMTP Host* y *SMTP Port* son los datos de acceso del servidor de correo electrónico desde el que se enviarán los correos de alerta. Los campos *User Authentication* y *User Password* son las credenciales de usuario para acceder al servidor de correo. El campo *From* es la dirección de la cuenta de correo desde la que se enviará el correo electrónico. El campo *To* es la dirección de correo electrónico del destinatario.

Cuando termine haga clic en **Save** para guardar los cambios. A partir de ese momento se enviarán correos de alerta a las direcciones especificadas cuando se produzcan eventos relevantes, como cuando se pierde la conexión con `altova.com`, por ejemplo. Estos eventos se registran en la pestaña **Messages**.

10.7.5 Mensajes, Cerrar sesión

La pestaña **Messages** muestra todos los mensajes relacionados con las licencias del repertorio de LicenseServer. Cada mensaje se puede eliminar haciendo clic en el correspondiente botón **Eliminar**.

La pestaña **Log out** es en realidad un botón de cierre de sesión. Al hacer clic en **Log out** se cierra inmediatamente la sesión y aparece la página de inicio de sesión.

© 2013 Altova GmbH

Índice

A

Altova LicenseServer,
ver LicenseServer, 378
Altova ServiceController, 383
API de Python,
API para XML, 137
API para XSD, 152
objeto Job, 136
Asignar licencias, 407, 417

C

Cargar licencias, 394, 413
Catálogos, 26
Catálogos XML, 26
Cerrar sesión, 424
Comando de ayuda de la ILC, 69
Comandos de licencias en la ILC, 70
Comandos XQuery, 63
Comandos XSLT, 58
Comprobar el formato, 51
Configuración, 421
Configuración del servidor, 98
Configurar, 14
Contraseña,
predeterminada al iniciar LicenseServer, 387
Contraseña predeterminada, 387
Correos de alerta, 421

D

Documentos XQuery,
validación, 66
Documentos XSLT,
validación, 61
Dot NET,
ver Interfaz .NET, 264

E

Ejecución de XQuery, 64
Ejemplo de script Python, 125, 129
Extensiones de Altova,
funciones de gráficos (ver funciones de gráficos), 335

F

FlowForce Server,
registrar con LicenseServer, 397
Funciones de extensión .NET,
constructores, 368
conversiones de tipos de datos (de .NET a XPath/XQuery), 372
conversiones de tipos de datos (de XPath/XQuery a .NET), 371
información general, 365
métodos de instancia, campos de instancia, 370
métodos estáticos, campos estáticos, 369
para XSLT y XQuery, 365
Funciones de extensión .NET para XSLT y XQuery,
ver Funciones de extensión .NET, 365
Funciones de extensión de Altova,
funciones de gráficos (ver funciones de gráficos), 336
funciones generales, 336
Funciones de extensión de scripts MSXSL, 373
Funciones de extensión Java,
archivos de clases definidos por el usuario, 356
archivos JAR definidos por el usuario, 359
constructores, 360
conversiones de tipos de datos (de Java a Xpath/XQuery), 364
conversiones de tipos de datos (de XPath/XQuery a Java), 363
información general, 354
métodos de instancia, campos de instancia, 362
métodos estáticos, campos estáticos, 361
para XSLT y XQuery, 354
Funciones de extensión Java para XSLT y XQuery,
ver Funciones de extensión Java, 354
Funciones de extensión para XSLT y XQuery, 334
Funciones de gráficos,
código de muestra, 342

© 2013 Altova GmbH

Funciones de gráficos,
ejemplo, 351
estructura de los datos del gráfico para, 346

I

Información de red, 379
Instalación,
en Linux, 16
en Mac OS X, 21
en Windows, 15
Instalación en Linux, 16
Instalación en Mac OS X, 21
Instalación en Windows, 15
Interfaces,
resumen, 5
Interfaz .NET, 5
uso, 264
Interfaz COM, 5
Interfaz del administrador, 412
Interfaz HTTP, 5, 92
configuración del servidor, 98
preparar el servidor, 94
problemas de seguridad, 35
solicitudes cliente, 102
Interfaz Java, 5
configuración, 216
documentación adicional, 216
uso, 216
Interfaz Python, 5, 120
crear scripts, 121
ejecutar scripts, 124

L

Licencias,
asignar, 407, 417
cargar, 394, 413
LicenseServer,
iniciar, 385
instalación en Linux, 381
instalación en Mac OS X, 382
instalación en Windows, 380
instrucciones para asignar licencias, 384
interfaz con, 412
opciones de configuración, 421
página de configuración, 412
registrar FlowForce Server con, 397
registrar MapForce Server con, 401
registrar StyleVision Server con, 405
Línea de comandos,
opciones, 74
resumen de uso, 38
y XQuery, 63
Localización, 71

M

MapForce Server,
registrar con LicenseServer, 401
Mensajes, 424
msxsl:script, 373

O

Opciones de configuración de red, 421

P

Página de configuración, 412
abrir en Linux, 390
abrir en Mac OS X, 392
abrir en Windows, 387
URL de, 387
URL de (Linux), 390
URL de (Mac OS X), 392
Página de configuración de LicenseServer,
(ver Página de configuración), 390, 392
ver Página de configuración, 387
Página Gestión de servidores, 417
Página Supervisión de servidores, 420
Pestaña Gestión de servidores, 407
Problemas de seguridad, 35
Python,
problemas de seguridad, 35

© 2013 Altova GmbH

R

RaptorXML,
características, 9
ediciones e interfaces, 5
especificaciones compatibles, 11
interfaces con COM, Java y .NET, 5
interfaz de la línea de comandos, 5
interfaz HTTP, 5
interfaz Python, 5
introducción, 3
requisitos del sistema, 8
Recursos globales, 33
Registrar FlowForce Server con LicenseServer, 397
Registrar MapForce Server con LicenseServer, 401
Registrar StyleVision Server con LicenseServer, 405
Repertorio de licencias, 394, 413

S

Scripts en XSLT/XQuery,
ver Funciones de extensión, 334
ServiceController, 383
StyleVision Server,
registrar con LicenseServer, 405

T

Transformación XSLT, 59

V

Validación,
de cualquier documento, 49
de documentos XQuery, 66
de documentos XSLT, 61
de DTD, 45
de instancias XML con DTD, 41
de instancias XML con XSD, 43
de XSD, 47

X

XQuery,
Funciones de extensión, 334
XSLT,
Funciones de extensión, 334

© 2013 Altova GmbH