Manual de referencia y del usuario

Manual del usuario y referencia de Altova RaptorXML+XBRL Server 2014

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: 2014

© 2014 Altova GmbH

Tabla de contenido

1 Introducción a RaptorXML+XBRL Server 3

1.1 Ediciones e interfaces ... 5
1.2 Requisitos del sistema ... 7
1.3 Características ... 8
1.4 Especificaciones compatibles ... 10

2 Instalar RaptorXML+XBRL Server 14

2.1 Instalación y configuración en Windows ... 15
2.1.1 Instalación en Windows ... 16
2.1.2 Asignación de licencias en Windows ... 19
2.2 Instalación y configuración en Linux ... 22
2.2.1 Instalación en Linux ... 23
2.2.2 Asignación de licencias en Linux ... 26
2.3 Instalación y configuración en Mac OS X ... 28
2.3.1 Instalación en Mac OS X ... 29
2.3.2 Asignación de licencias en Mac OS X ... 32
2.4 Catálogos XML ... 34
2.4.1 Cómo funcionan los catálogos ... 35
2.4.2 Mecanismo de catalogación XML de Altova ... 37
2.4.3 Variables para ubicaciones de sistemas Windows ... 40
2.5 Recursos globales ... 42
2.6 Problemas de seguridad ... 44

3 Interfaz de la línea de comandos (ILC) 46

3.1 Comandos para validar XML, DTD, XSD ... 49
3.1.1 valxml-withdtd (xml) ... 50
3.1.2 valxml-withxsd (xsi) ... 54
3.1.3 valdtd (dtd) ... 60
3.1.4 valxsd (xsd) ... 63
3.1.5 valany ... 68
3.2 Comandos para comprobar el formato ... 72
3.2.1 wfxml ... 73

3.2.2 wfdtd ... 76
3.2.3 wfany ... 79
3.3 Comandos de validación XBRL ... 82
3.3.1 valxbrl (xbrl) ... 83
3.3.2 valxbrltaxonomy (dts) ... 92
3.3.3 valany ... 98
3.4 Comandos XSLT ... 102
3.4.1 xslt ... 103
3.4.2 valxslt ... 109
3.5 Comandos XQuery ... 114
3.5.1 xquery ... 115
3.5.2 valxquery ... 120
3.6 Comandos de ayuda y licencias ... 123
3.6.1 Ayuda ... 124
3.6.2 Licencias ... 125
3.7 Comandos de localización ... 127
3.7.1 exportresourcestrings ... 128
3.7.2 setdeflang ... 129
3.8 Opciones ... 130
3.8.1 Catálogos, recursos globales, archivos ZIP ... 131
3.8.2 Mensajes, errores y ayuda ... 132
3.8.3 Procesamiento ... 133
3.8.4 XBRL ... 134
3.8.5 XML ... 141
3.8.6 XSD ... 142
3.8.7 XQuery ... 144
3.8.8 XSLT ... 146

4 Interfaz HTTP 150

4.1 Preparar el servidor ... 152
4.1.1 Iniciar el servidor ... 153
4.1.2 Probar la conexión ... 155
4.1.3 Configurar el servidor ... 156
4.2 Solicitudes cliente ... 160
4.2.1 Iniciar trabajos con POST ... 163
4.2.2 Respuesta del servidor a solicitudes POST ... 168
4.2.3 Obtener el documento de resultados ... 171
4.2.4 Obtener los documentos de salida/errores ... 175
4.2.5 Liberar espacio tras el procesamiento ... 177

5 Interfaz Python 180

5.1 Crear scripts Python ... 182
5.2 Ejecutar scripts Python ... 185
5.3 Ejemplo de script 1: procesar XML ... 186
5.3.1 Script ... 187
5.3.2 Documento de resultados ... 190
5.4 Ejemplo de script 2: cambiar el formato XML ... 191
5.4.1 Script ... 192
5.4.2 Documento de resultados ... 195
5.5 Ejemplo de script 3: informe XBRL ... 198
5.5.1 Script ... 199
5.5.2 Documento de resultados ... 202
5.6 API de Python: el objeto Job ... 204
5.7 API de Python para XML ... 205
5.7.1 xml.Attribute ... 207
5.7.2 xml.Character ... 208
5.7.3 xml.Comment ... 209
5.7.4 xml.Document ... 210
5.7.5 xml.Element ... 211
5.7.6 xml.Namespace ... 213
5.7.7 xml.Notation ... 214
5.7.8 xml.NSAttribute ... 215
5.7.9 xml.ProcessingInstruction ... 216
5.7.10 xml.QName ... 217
5.7.11 xml.UnexpandedEntityReference ... 218
5.7.12 xml.UnparsedEntity ... 219
5.8 API de Python para XSD ... 220
5.8.1 xsd.Annotation ... 225
5.8.2 xsd.Any ... 226
5.8.3 xsd.AnyAttribute ... 227
5.8.4 xsd.Assertion ... 228
5.8.5 xsd.AttributeDeclaration ... 229
5.8.6 xsd.AttributeGroupDefinition ... 230
5.8.7 xsd.AttributePSVI ... 231
5.8.8 xsd.AttributeUse ... 233
5.8.9 xsd.Block ... 234
5.8.10 xsd.ComplexTypeDefnition ... 235
5.8.11 xsd.ContentType ... 236

5.8.12 xsd.Defined ... 237
5.8.13 xsd.DerivationMethod ... 238
5.8.14 xsd.ENTITY ... 239
5.8.15 xsd.ElementDeclaration ... 240
5.8.16 xsd.ElementPSVI ... 242
5.8.17 xsd.Final ... 244
5.8.18 xsd.ID ... 245
5.8.19 xsd.IDREF ... 246
5.8.20 xsd.ID_IDREF_binding ... 247
5.8.21 xsd.ID_IDREF_table ... 248
5.8.22 xsd.IdentityConstraintDefinition ... 249
5.8.23 xsd.Instance ... 250
5.8.24 xsd.ModelGroup ... 251
5.8.25 xsd.ModelGroupDefinition ... 252
5.8.26 xsd.NCName ... 253
5.8.27 xsd.NMTOKEN ... 254
5.8.28 xsd.NOTATION ... 255
5.8.29 xsd.Name ... 256
5.8.30 xsd.NamespaceBinding ... 257
5.8.31 xsd.NamespaceConstraint ... 258
5.8.32 xsd.NotationDeclaration ... 259
5.8.33 xsd.OpenContent ... 260
5.8.34 xsd.PSVI ... 261
5.8.35 xsd.Particle ... 262
5.8.36 xsd.QName ... 263
5.8.37 xsd.Schema ... 264
5.8.38 xsd.Scope ... 266
5.8.39 xsd.Sibling ... 267
5.8.40 xsd.SimpleTypeDefinition ... 268
5.8.41 xsd.TypeAlternative ... 270
5.8.42 xsd.TypeTable ... 271
5.8.43 xsd.Unbounded ... 272
5.8.44 xsd.ValueConstraint ... 273
5.8.45 xsd.XPathExpression ... 274
5.8.46 Objetos de tipos de datos especiales integrados ... 275
5.8.47 Objetos de tipo de datos string ... 276
5.8.48 Objeto de tipo de datos boolean ... 277
5.8.49 Objetos de tipo de datos number ... 278
5.8.50 Objetos de tipo de datos duration ... 279
5.8.51 Objetos de tipo de datos date y time ... 280
5.8.52 Objetos de tipo de datos binary ... 281

5.8.53 Objetos faceta ... 282
5.9 API de Python para XBRL ... 284
5.9.1 xbrl.BreakdownResource ... 287
5.9.2 xbrl.Concept ... 288
5.9.3 xbrl.ConceptAspectValue ... 294
5.9.4 xbrl.ConstraintSet ... 295
5.9.5 xbrl.Context ... 298
5.9.6 xbrl.DefinitionNodeResource ... 299
5.9.7 xbrl.DTS ... 300
5.9.8 xbrl.Entity ... 301
5.9.9 xbrl.EntityIdentifier ... 302
5.9.10 xbrl.EntityIdentifierAspectValue ... 303
5.9.11 xbrl.Error ... 304
5.9.12 xbrl.ExplicitDimensionAspectValue ... 305
5.9.13 xbrl.Fact ... 306
5.9.14 xbrl.FactSet ... 309
5.9.15 xbrl.Fraction ... 310
5.9.16 xbrl.Instance ... 311
5.9.17 xbrl.LabelResource ... 313
5.9.18 xbrl.LayoutCell ... 314
5.9.19 xbrl.LayoutDataCell ... 315
5.9.20 xbrl.LayoutHeaderCell ... 316
5.9.21 xbrl.LayoutRow ... 317
5.9.22 xbrl.LayoutTable ... 318
5.9.23 xbrl.LayoutTableSet ... 319
5.9.24 xbrl.Period ... 320
5.9.25 xbrl.PeriodAspectValue ... 322
5.9.26 xbrl.ReferenceResource ... 323
5.9.27 xbrl.Resource ... 324
5.9.28 xbrl.ScenarioAspectValue ... 325
5.9.29 xbrl.SegmentAspectValue ... 326
5.9.30 xbrl.TableError ... 327
5.9.31 xbrl.TableResource ... 328
5.9.32 xbrl.TypedDimensionAspectValue ... 329
5.9.33 xbrl.Unit ... 330
5.9.34 xbrl.UnitAspectValue ... 331

6 Interfaz Java 334

6.1 Ejemplo de proyecto Java ... 336
6.2 Interfaces de RaptorXML para Java ... 338

6.2.1 RaptorXMLFactory ... 339
6.2.2 XMLValidator ... 346
6.2.3 XSLT ... 357
6.2.4 XQuery ... 365
6.2.5 XBRL ... 372
6.2.6 RaptorXMLException ... 382

7 Interfaces COM y .NET 384

7.1 Notas sobre la interfaz COM ... 385
7.2 Notas sobre la interfaz NET ... 386
7.3 Lenguajes de programación ... 388
7.3.1 Ejemplo de COM: VBScript ... 390
7.3.2 Ejemplo de .NET: C# ... 393
7.3.3 Ejemplo de .NET: Visual Basic .NET ... 396
7.4 Referencia de la API ... 398
7.4.1 Interfaces ... 399
IServer ... 399
IXMLValidator ... 403
IXSLT ... 407
IXQuery ... 413
IXBRL ... 417
7.4.2 Enumeraciones ... 425
ENUMAssessmentMode ... 425
ENUMErrorFormat ... 426
ENUMLoadSchemalocation ... 426
ENUMQueryVersion ... 427
ENUMSchemaImports ... 428
ENUMSchemaMapping ... 429
ENUMTableOutputFormat ... 430
ENUMValidationType ... 430
ENUMWellformedCheckType ... 431
ENUMXBRLValidationType ... 432
ENUMXMLValidationMode ... 433
ENUMXQueryVersion ... 434
ENUMXSDVersion ... 434
ENUMXSLTVersion ... 435

8 Información adicional 438

8.1 Sugerencias sobre la ubicación del esquema ... 439

8.2 Parámetros de XBRL Formula ... 440
8.2.1 Formatos de los parámetros de fórmula ... 441
8.2.2 Usar parámetros de fórmula ... 443

9 Información sobre motores XSLT y XQuery 448

9.1 XSLT 1.0 ... 449
9.2 XSLT 2.0 ... 450
9.3 XSLT 3.0 ... 453
9.4 XQuery 1.0 ... 454
9.5 XQuery 3.0 ... 458

10 Funciones XSTL y XPath/XQuery 460

10.1 Funciones de extensión de Altova ... 462
10.1.1 Funciones XSLT ... 464
10.1.2 Funciones XPath/XQuery: fecha y hora ... 467
10.1.3 Funciones XPath/XQuery: funciones de cadena ... 477
10.1.4 Funciones XPath/XQuery: funciones varias ... 481
10.1.5 Funciones para gráficos ... 486
Estructura XML de los datos de gráficos ... 490
Ejemplo: funciones para gráficos ... 495
10.1.6 Funciones para códigos de barras ... 500
10.2 Funciones de extensión varias ... 503
10.2.1 Funciones de extensión Java ... 504
Archivos de clases definidos por el usuario ... 505
Archivos JAR definidos por el usuario ... 508
Constructores ... 509
Métodos estáticos y campos estáticos ... 510
Métodos de instancia y campos de instancia ... 510
Tipos de datos: conversión de XPath/XQuery en Java ... 511
Tipos de datos: conversión de Java en XPath/XQuery ... 512
10.2.2 Funciones de extensión .NET ... 513
Constructores ... 515
Metodos estáticos y campos estáticos ... 516
Métodos de instancia y campos de instancia ... 517
Tipos de datos: conversión de XPath/XQuery en .NET ... 518
Tipos de datos: conversión de .NET en XPath/XQuery ... 519
10.2.3 Funciones XBRL para XSLT ... 520
10.2.4 Scripts MSXSL para XSLT ... 521

11 Altova LicenseServer 526

11.1 Información de red ... 528
11.2 Instalación en Windows ... 529
11.3 Instalación en Linux ... 530
11.4 Instalación en Mac OS X ... 532
11.5 Altova ServiceController ... 533
11.6 Asignación de licencias ... 534
11.6.1 Iniciar LicenseServer ... 535
11.6.2 Abrir la página de configuración de LicenseServer (Windows) ... 537
11.6.3 Abrir la página de configuración de LicenseServer (Linux) ... 540
11.6.4 Abrir la página de configuración de LicenseServer (Mac OS X) ... 542
11.6.5 Cargar licencias en LicenseServer ... 544
11.6.6 Registrar FlowForce Server con LicenseServer ... 547
11.6.7 Registrar MapForce Server con LicenseServer ... 551
11.6.8 Registrar StyleVision Server con LicenseServer ... 553
11.6.9 Registrar RaptorXML(+XBRL) Server con LicenseServer ... 555
11.6.10 Asignar licencias a productos registrados ... 557
11.7 Referencia de la página de configuración ... 562
11.7.1 Repertorio de licencias ... 563
11.7.2 Gestión de servidores ... 567
11.7.3 Supervisión de servidores ... 571
11.7.4 Opciones de configuración ... 572
11.7.5 Mensajes, Cerrar sesión ... 576

Altova RaptorXML+XBRL Server 2014

Introducción a RaptorXML+XBRL Server

1 Introducción a RaptorXML+XBRL Server

Altova RaptorXML+XBRL 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).

Ediciones y sistemas operativos
Altova ofrece dos ediciones diferentes de RaptorXML, diseñadas para satisfacer diferentes requisitos. Estas dos ediciones se describen en el apartado Ediciones e interfaces. RaptorXML está disponible 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 y XBRL, 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)
- Instalar 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: 23/04/2014

© 2014 Altova GmbH

1.1 Ediciones e interfaces

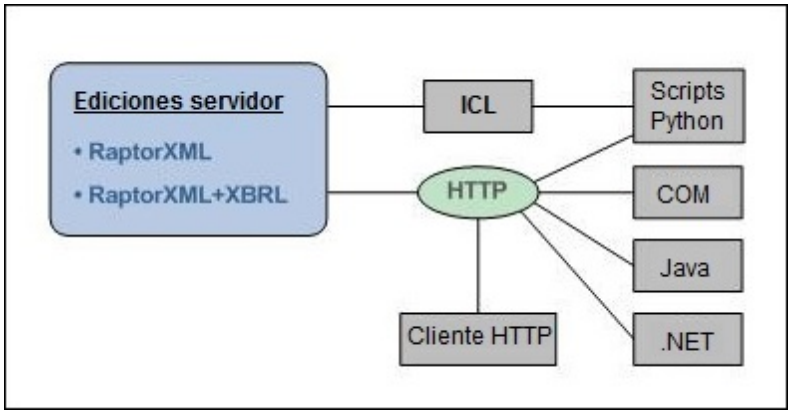
Altova ofrece dos ediciones distintas de RaptorXML:

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

Interfaces
Puede acceder a RaptorXML+XBRL Server a través de varias interfaces:

- Una **interfaz de la línea de comandos** (en adelante ***ILC***).
- Una **interfaz COM** para sistemas Windows.
- Una **interfaz .NET** para sistemas Windows.
- Una **interfaz Java** para sistemas Windows, Linux y Mac OS.
- Una **interfaz HTTP** a la que se puede acceder desde un cliente HTTP.
- 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.

El diagrama que aparece a continuación muestra cómo se accede a RaptorXML Server y RaptorXML+XBRL Server a través de las diferentes interfaces.



Observe que las interfaces COM, Java y .NET usan el protocolo HTTP para conectarse a RaptorXML. Los scripts Python se pueden enviar a RaptorXML 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 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.

 © 2014 Altova GmbH

1.2 Requisitos del sistema

RaptorXML+XBRL Server es compatible con estos sistemas operativos:

Windows Server
Windows Server 2008 R2 o superior

Windows
Windows XP con Service Pack 3, Windows 7, Windows 8 o superior

Linux
- CentOS 6 o superior
- RedHat 6 o superior
- Debian 6 o superior
- Ubuntu 12.04 o superior

Mac OS X
Mac OS X 10.7 o superior

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

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

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, XBRL, DTD, esquemas XML, XSLT o XQuery).

Validación XML y XBRL

- Valida documentos XML y XBRL con esquemas XML y DTD internos o externos.
- Revisa el formato de documentos XML, DTD, XML Schema, XSLT y XQuery.
- Valida taxonomías XBRL y documentos XBRL con taxonomías XBRL.
- Ejecuta fórmulas XBRL y aserciones de validación.
- Compatible con las especificaciones XBRL 2.1, Dimensions 1.0 y Formula 1.0 y con la propuesta de recomendación Table Linkbase 1.0 del 18 de diciembre de 2013.

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 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

 © 2014 Altova GmbH

- 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.

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

Recomendaciones XBRL
Sitio web: Extensible Business Reporting Language (XBRL)

- XBRL 2.1
- Dimensions 1.0
- Formula Specification 1.0
  - Filtros de aspecto cover

 © 2014 Altova GmbH

  - Filtros booleanos
  - Filtros de concepto
  - Filtros de relación de conceptos
  - Aserciones de coherencia
  - Implementación de funciones personales
  - Filtros de dimensión
  - Filtros de entidad
  - Aserciones de existencia
  - Fórmula
  - Registro de funciones
  - Mensajes genéricos
  - Referencias genéricas
  - Filtros implicit
  - Filtros match
  - Filtros de período
  - Filtros relativos
  - Filtros de segmento
  - Filtros de tupla
  - Filtros de unidad
  - Validación
  - Mensajes de validación
  - Aserciones de valor
  - Filtros de valor
  - Variables
- Table Linkbase 1.0 (propuesta de recomendación del 18 de diciembre de 2013)
- Function Definitions
- Generic Links 1.0
  - Filtros generales
  - Etiquetas genéricas

Altova RaptorXML+XBRL Server 2014

Instalar RaptorXML+XBRL Server

2 Instalar RaptorXML+XBRL Server

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

- Cómo instalar RaptorXML y asignarle licencias en sistemas Windows, Linux y 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.

 © 2014 Altova GmbH

2.1 Instalación y configuración en Windows

Esta sección explica cómo instalar RaptorXML+XBRL Server y asignarle licencias en sistemas Windows.

Instalación en Windows

- Requisitos del sistema
- Instalación e instaladores
- Instalar Altova LicenseServer
- Ubicación de la carpeta de aplicación

Asignación de licencias en Windows

- Cómo asignar licencias
- Iniciar LicenseServer como servicio
- Registrar RaptorXML+XBRL Server
- Nota sobre núcleos y licencias

2.1.1 Instalación en Windows

Temas de este apartado:

- Requisitos del sistema
- Instaladores e instalación
- Idioma
- Instalar Altova LicenseServer
- Licencia de prueba
- Versiones de LicenseServer y compatibilidad con RaptorXML+XBRL Server
- Carpeta de la aplicación

Requisitos del sistema

Windows Server
Windows Server 2008 R2 o superior

Windows
Windows XP con Service Pack 3, Windows 7, Windows 8 o superior

---

Instaladores e instalación

Desde el sitio web de Altova puede descargar el instalador de RaptorXML+XBRL Server, que instala RaptorXML+XBRL Server y realiza los registros pertinentes. El ejecutable de RaptorXML +XBRL Server se guarda por defecto en esta carpeta:

```
<CarpetaArchivosPrograma>\Altova\RaptorXMLXBRLServer2014\bin\RaptorXMLXBRL.exe
```

El instalador realiza todos los registros necesarios para usar RaptorXML+XBRL Server desde la interfaz de COM, como interfaz Java y en el entorno .NET. Esto también incluye registrar el ejecutable deRaptorXML+XBRL Server como objeto servidor COM, instalar `RaptorXMLLib.dll` (para su uso en la interfaz Java) en el directorio `WINDIR\system32\` y añadir el archivo `Altova.RaptorXML.dll` a la biblioteca de referencia de .NET.

Idioma

Puede seleccionar el idioma del asistente para la instalación en el cuadro combinado situado en la parte inferior izquierda del asistente. Los idiomas disponibles actualmente son español, inglés, alemán y japonés. El idioma que seleccione aquí también será el idioma de la interfaz del usuario de RaptorXML+XBRL Server.

Instalar Altova LicenseServer

Para poder ejecutar los productos servidores de Altova, incluido RaptorXML+XBRL Server, es necesario asignarles una licencia con un servidor Altova LicenseServer de la red.

 © 2014 Altova GmbH

El instalador de FlowForce Server y de RaptorXML+XBRL Server para sistemas Windows incluye una opción para instalar Altova LicenseServer. Si ya hay un servidor Altova LicenseServer en su red, no es necesario instalarlo otra vez. Durante el proceso de instalación de FlowForce Server o de RaptorXML+XBRL Server, debe activar/desactivar la casilla que permite instalar Altova LicenseServer.

Para más información sobre cómo registrar RaptorXML+XBRL Server con el servidor Altova LicenseServer y asignarle licencias consulte el apartado Asignación de licencias en Windows.

Licencia de prueba
Durante el proceso de instalación tiene la opción de solicitar gratis una licencia de prueba de 30 días para RaptorXML+XBRL Server. Una vez enviada la solicitud, Altova le enviará una licencia de prueba a la dirección de correo electrónico indicada.

Versiones de LicenseServer y compatibilidad con RaptorXML+XBRL Server
Cuando actualice RaptorXML+XBRL Server a una versión más reciente, asegúrese de actualizar también su servidor LicenseServer con la versión más reciente. Las versiones anteriores de RaptorXML+XBRL Server funcionarán con las conversiones más recientes de LicenseServer.

Es decir, si instala una versión nueva de RaptorXML+XBRL Server y la versión actual de su servidor LicenseServer no es la más reciente, desinstale la versión antigua de LicenseServer e instale la versión más reciente (disponible en el sitio web de Altova). Toda la información de registro y licencias almacenada en la versión antigua se guardará en una base de datos su equipo servidor durante la desinstalación y se importará automáticamente a la versión nueva.

El número de versión de su servidor LicenseServer actual aparece en la parte inferior de la página de configuración. El número de versión de LicenseServer también aparece durante la instalación (tanto de LicenseServer como de RaptorXML+XBRL Server). Recuerde que puede instalar LicenseServer junto con RaptorXML+XBRL Server o por separado. Si quiere actualizar LicenseServer con una versión más reciente, recuerde que antes debe desinstalar la versión anterior.

Ubicación de la carpeta de aplicación
La aplicación se instala en esta carpeta:

Windows 8, Windows 7	`C:\Archivos de programa\Altova\`
Windows XP, Windows Vista	`C:\Archivos de programa\Altova\`

Versión de 32 bits en un sistema operativo de 64 bits	`C:\Archivos de programa (x86)\Altova\`

© 2014 Altova GmbH

2.1.2 Asignación de licencias en Windows

Temas de este apartado:

- Cómo asignar licencias
- Iniciar LicenseServer como servicio
- Registrar RaptorXML+XBRL Server
- Nota sobre núcleos y licencias

Cómo asignar licencias al producto
Siga estos pasos para asignar una licencia a RaptorXML+XBRL Server en un sistema Windows con un servidor Altova LicenseServer instalado en un equipo de la red:

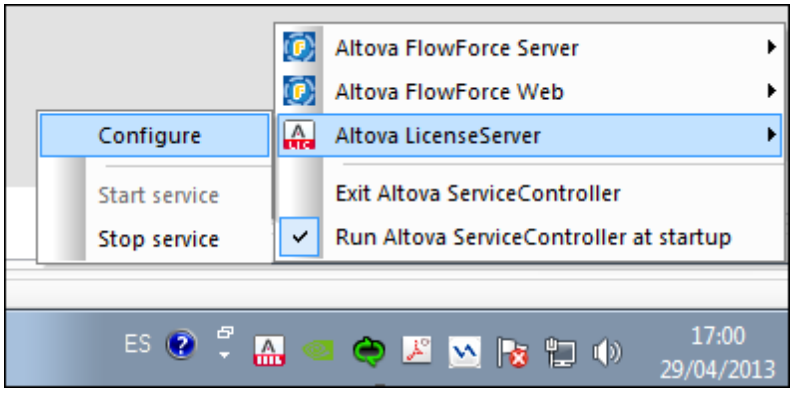
1. Si todavía no está ejecutándose, inicie Altova LicenseServer como servicio.
2. Registre RaptorXML+XBRL Server con LicenseServer desde la línea de comandos de RaptorXML+XBRL Server. Recuerde que RaptorXML+XBRL Server se puede registrar automáticamente con LicenseServer si marca la casilla correspondiente durante el proceso de instalación.
3. En la pestaña **Server Management** de la página de configuración de LicenseServer, asigne una licencia a RaptorXML+XBRL Server.

Nota: el instalador de RaptorXML+XBRL Server incluye una opción para instalar Altova LicenseServer.

Nota: durante el proceso de instalación tiene la opción de solicitar gratis una licencia de evaluación de 30 días.

Iniciar LicenseServer como servicio
Puede iniciar LicenseServer desde el controlador de servicios Altova ServiceController, que está en la bandeja del sistema.

Primero 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 (*imagen siguiente*). Si selecciona la opción **Run Altova ServiceController at Startup**, cuando inicie sesión en el sistema Altova ServiceController se iniciará 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 anterior*) y seleccione **Start Service** para iniciar el servicio. Si LicenseServer ya está en ejecución, la opción **Start Service** aparece deshabilitada (*imagen anterior*).

Registrar RaptorXML+XBRL Server

RaptorXML+XBRL Server debe registrarse con un servidor Altova LicenseServer de su red o de su equipo y se le debe asignar una licencia desde la interfaz gráfica de ese servidor de licencias. Puede registrar RaptorXML+XBRL Server desde su interfaz de la línea de comandos con el comando `licenseserver`:

```
RaptorXMLXBRL licenseserver [opciones] NombreServidor-O-Dirección-IP
```

Por ejemplo, si el nombre del servidor donde está instalado LicenseServer es `localhost`, entonces puede registrar RaptorXML+XBRL Server desde su interfaz de la línea de comandos con el LicenseServer que está en `localhost`.

```
RaptorXMLXBRL licenseserver localhost
```

Tras registrarse, RaptorXML+XBRL Server aparece en la lista de la pestaña **Server Management** de la página de configuración de LicenseServer, donde puede asignarle una licencia.

Nota sobre núcleos y licencias

La asignación de licencias para RaptorXML+XBRL Server depende del número de núcleos disponibles en el equipo donde se ejecuta RaptorXML+XBRL 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.

2.2 Instalación y configuración en Linux

Esta sección explica cómo instalar RaptorXML+XBRL Server y asignarle licencias en sistemas Linux (Debian, Ubuntu, CentOS, RedHat).

Instalación en Linux

- Requisitos del sistema
- Nota sobre el usuario raíz
- Desinstalar versiones previas de productos servidor de Altova
- Descargar el paquete de instalación para Linux
- Instalar RaptorXML+XBRL Server
- Instalar Altova LicenseServer

Asignación de licencias en Linux

- Cómo asignar licencias
- Nota sobre el usuario raíz
- Iniciar LicenseServer como servicio
- Registrar RaptorXML+XBRL Server
- Nota sobre núcleos y licencias

 © 2014 Altova GmbH

2.2.1 Instalación en Linux

Temas de este apartado:

- Requisitos del sistema
- Nota sobre el usuario raíz
- Desinstalar versiones previas de productos servidor de Altova
- Descargar el paquete de instalación para Linux
- Instalar RaptorXML+XBRL Server
- Instalar Altova LicenseServer
- Versiones de LicenseServer y compatibilidad con RaptorXML+XBRL Server

---

Requisitos del sistema

Linux

- CentOS 6 o superior
- RedHat 6 o superior
- Debian 6 o superior
- Ubuntu 12.04 o superior

Nota sobre el usuario raíz

Para instalar RaptorXML+XBRL Server es necesario tener privilegios de administrador (`root`). Por tanto, debe realizar la instalación como usuario raíz. Si inició sesión como `root`, puede ahorrarse la palabra clave `sudo` de los comandos que aparecen a continuación.

Desinstalar versiones previas de productos servidor de Altova

En la interfaz de la línea de comandos de Linux puede comprobar si ya hay productos servidor de Altova instalados en el equipo. Para ello use este comando:

```
[Debian, Ubuntu]:   dpkg --list | grep Altova

[CentOS, RedHat]:   rpm -qa | grep server
```

Si RaptorXML+XBRL Server no está instalado, continúe con la instalación tal y como se describe en el apartado Instalar RaptorXML+XBRL Server.

Si RaptorXML+XBRL Server ya está instalado y quiere instalar una versión nueva, antes debe desinstalar la versión previa con este comando:

```
[Debian, Ubuntu]:   sudo dpkg --remove raptorxmlxbrlserver

[CentOS, RedHat]:   sudo rpm -e raptorxmlxbrlserver
```

Si quiere desinstalar una versión previa de Altova LicenseServer, use este comando:

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

Descargar el paquete de instalación para Linux
Los paquetes de instalación de RaptorXML+XBRL Server para sistemas Linux se pueden descargar del sitio web de Altova.

Distribución	Extensión del paquete
Debian 6 y superior	`.deb`
Ubuntu12.04 y superior	`.deb`
CentOS 6 y superior	`.rpm`
RedHat 6 y superior	`.rpm`

Tras descargarlo, copie el paquete de instalación en cualquier directorio del sistema Linux. Para ejecutar RaptorXML+XBRL Server es necesario tener instalado Altova LicenseServer, que también se puede descargar del sitio web de Altova.

Instalar RaptorXML+XBRL Server
En una ventana de la Terminal, cambie al directorio donde copió el paquete de instalación para 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 RaptorXML+XBRL Server con este comando:

```
[Debian]:   sudo dpkg --install raptorxmlxbrlserver-2014-debian.deb
[Ubuntu]:   sudo dpkg --install raptorxmlxbrlserver-2014-ubuntu.deb
[CentOS]:   sudo rpm -ivh raptorxmlxbrlserver-2014-1.x86_64.rpm
[RedHat]:   sudo rpm -ivh raptorxmlxbrlserver-2014-1.x86_64.rpm
```

La aplicación RaptorXML+XBRL Server se instala en este directorio:

```
/opt/Altova/RaptorXMLXBRLServer2014
```

Instalar Altova LicenseServer
Para poder ejecutar los productos servidores de Altova, incluido RaptorXML+XBRL Server, es

 © 2014 Altova GmbH

necesario asignarles una licencia con un servidor Altova LicenseServer de la red.

En los sistemas Linux es necesario instalar Altova LicenseServer por separado. Por tanto, descargue Altova LicenseServer del sitio web de Altova y copie el paquete de instalación en cualquier directorio. Siga las instrucciones anteriores para instalar LicenseServer (*ver paso anterior*).

```
[Debian]:   sudo dpkg --install licenseserver-1.8-debian.deb
[Ubuntu]:   sudo dpkg --install licenseserver-1.8-ubuntu.deb
[CentOS]:   sudo rpm -ivh licenseserver-1.8-1.x86_64.rpm
[RedHat]:   sudo rpm -ivh licenseserver-1.8-1.x86_64.rpm
```

La aplicación LicenseServer se instala en este directorio:

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

Consulte el apartado siguiente Asignación de licencias en Linux para obtener información sobre cómo registrar RaptorXML+XBRL Server con Altova LicenseServer y asignarle licencias.

Versiones de LicenseServer y compatibilidad con RaptorXML+XBRL Server
Cuando actualice RaptorXML+XBRL Server a una versión más reciente, asegúrese de actualizar también su servidor LicenseServer con la versión más reciente. Las versiones anteriores de RaptorXML+XBRL Server funcionarán con las conversiones más recientes de LicenseServer.

Es decir, si instala una versión nueva de RaptorXML+XBRL Server y la versión actual de su servidor LicenseServer no es la más reciente, desinstale la versión antigua de LicenseServer e instale la versión más reciente (disponible en el sitio web de Altova). Toda la información de registro y licencias almacenada en la versión antigua se guardará en una base de datos su equipo servidor durante la desinstalación y se importará automáticamente a la versión nueva.

2.2.2 Asignación de licencias en Linux

Temas de este apartado:

- Cómo asignar licencias
- Nota sobre el usuario raíz
- Iniciar LicenseServer como servicio
- Registrar RaptorXML+XBRL Server
- Nota sobre núcleos y licencias

Cómo asignar licencias

Siga estos pasos para asignar una licencia a RaptorXML+XBRL Server en sistemas Linux (Debian, Ubuntu, CentOS, RedHat) con un servidor Altova LicenseServer instalado en un equipo de la red:

1. Si Altova LicenseServer todavía no está ejecutándose, inicie Altova LicenseServer como servicio.
2. Registre RaptorXML+XBRL Server con LicenseServer.
3. En la pestaña **Server Management** de la página de configuración de LicenseServer, asigne una licencia a RaptorXML+XBRL Server.

Nota sobre el usuario raíz

Para registrar RaptorXML+XBRL Server con LicenseServer es necesario tener privilegios de administrador (`root`). Por tanto, debe realizar la asignación de licencias como usuario raíz. Si inició sesión como `root`, puede ahorrarse la palabra clave `sudo` de los comandos que aparecen a continuación.

Iniciar LicenseServer como servicio

Para registrar RaptorXML+XBRL Server y asignarle una licencia correctamente, debe estar en ejecución como servicio. Inicie LicenseServer como servicio con este comando:

```
[Debian]:   sudo /etc/init.d/licenseserver start

[Ubuntu]:   sudo initctl start licenseserver

[CentOS]:   sudo initctl start licenseserver

[RedHat]:   sudo initctl start licenseserver
```

Si por cualquier motivo necesita detener LicenseServer, use el mismo comando pero sustituya **`stop`** por **`start`**. Por ejemplo:

```
sudo /etc/init.d/licenseserver stop
```

 © 2014 Altova GmbH

Registrar RaptorXML+XBRL Server
Antes de poder asignarle una licencia, RaptorXML+XBRL Server debe estar registrado con LicenseServer. Puede usar el comando `licenseserver` de la interfaz de la línea de comandos para registrar RaptorXML+XBRL Server:

```
sudo /opt/Altova/RaptorXMLXBRLServer2014/bin/raptorxmlxbrl licenseserver
[opciones] NombreServidor-O-Dirección-IP
```

Por ejemplo:

```
sudo /opt/Altova/RaptorXMLXBRLServer2014/bin/raptorxmlxbrl licenseserver
localhost
```

En el ejemplo anterior `localhost` es el nombre del servidor donde está instalado LicenseServer. Observe además que el ejecutable de RaptorXML+XBRL Server está en esta ubicación:

```
/opt/Altova/RaptorXMLXBRLServer2014/bin
```

Tras registrarse, RaptorXML+XBRL Server aparece en la lista de la pestaña **Server Management** de la página de configuración de LicenseServer, donde puede asignarle una licencia.

Nota sobre núcleos y licencias
La asignación de licencias para RaptorXML+XBRL Server depende del número de núcleos disponibles en el equipo donde se ejecuta RaptorXML+XBRL 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.

2.3 Instalación y configuración en Mac OS X

Esta sección explica cómo instalar RaptorXML+XBRL Server y asignarle licencias en sistemas Mac OS X.

Instalación en Mac OS X

- Requisitos del sistema
- Nota sobre el usuario raíz
- Desinstalar versiones previas de productos servidor de Altova
- Descargar el paquete de instalación para Mac OS X
- Instalar RaptorXML+XBRL Server
- Instalar Altova LicenseServer

Asignación de licencias en Mac OS X

- Cómo asignar licencias
- Nota sobre el usuario raíz
- Iniciar LicenseServer como servicio
- Iniciar RaptorXML+XBRL Server como servicio
- Registrar RaptorXML+XBRL Server
- Nota sobre núcleos y licencias

 © 2014 Altova GmbH

2.3.1 Instalación en Mac OS X

Temas de este apartado:

- Requisitos del sistema
- Nota sobre el usuario raíz
- Desinstalar versiones previas de productos servidor de Altova
- Descargar el paquete de instalación para Mac OS X
- Instalar RaptorXML+XBRL Server
- Instalar Altova LicenseServer
- Versiones de LicenseServer y compatibilidad con RaptorXML+XBRL Server

---

Requisitos del sistema

Mac OS X
Mac OS X 10.7 o superior

Nota sobre el usuario raíz
Para instalar RaptorXML+XBRL Server en Mac OS X es necesario tener privilegios de administrador (usuario raíz "root"). Por tanto, debe realizar la instalación como usuario raíz.

Desinstalar versiones previas de productos servidor de Altova
Si necesita desinstalar versiones previas del producto servidor de Altova, siga las instrucciones que aparecen a continuación.

Antes de desinstalar RaptorXML+XBRL Server, detenga el servicio con este comando:

```
sudo launchctl unload /Library/LaunchDaemons/
com.altova.RaptorXMLXBRLServer2014.plist
```

Compruebe si se detuvo el servicio: abra la terminal del Monitor de actividad y compruebe que RaptorXML+XBRL Server no está en la lista.

En la terminal de Aplicaciones haga clic con el botón secundario en el icono de RaptorXML +XBRL 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/RaptorXMLXBRLServer2014/
```

Si necesita desinstalar una versión antigua de Altova LicenseServer, antes debe detener el servicio.

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

Para comprobar si el servicio se detuvo o no, abra la terminal del Monitor de actividad y confirme que LicenseServer no está en la lista. Después desinstale LicenseServer siguiendo las instrucciones dadas más arriba para RaptorXML+XBRL Server.

Descargar el paquete de instalación para Mac OS X
Tras descargarlo del sitio web Altova, copie el paquete de instalación en cualquier directorio del sistema Mac OS X. Para poder ejecutar RaptorXML+XBRL Server es necesario tener instalado Altova LicenseServer. Por tanto, descargue LicenseServer del sitio web de Altova junto con el paquete de instalación de RaptorXML+XBRL Server. El archivo del instalador para Mac OS X tiene la extensión `.pkg`.

Instalar RaptorXML+XBRL Server
En una ventana de la Terminal cambie el directorio donde copió el archivo del instalador y haga doble clic en él. Siga los pasos del asistente para la instalación y acepte el contrato de licencia.

La aplicación RaptorXML+XBRL Server se instala en esta carpeta:

```
/usr/local/Altova/RaptorXMLXBRLServer2014
```

Haga clic en el icono de RaptorXML+XBRL Server en la terminal de Aplicaciones para ver la ayuda en pantalla (es decir, la presente documentación).

Instalar Altova LicenseServer
Para poder ejecutar los productos servidores de Altova, incluido RaptorXML+XBRL Server, es necesario asignarles una licencia con un servidor Altova LicenseServer de la red.

En los sistemas Mac OS X es necesario instalar Altova LicenseServer por separado. Por tanto, descargue Altova LicenseServer del sitio web de Altova y haga doble clic en el instalador para iniciar la instalación. Siga las instrucciones en pantalla y acepte el contrato de licencia.

Altova LicenseServer se instala en esta carpeta:

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

Para más información sobre cómo registrar RaptorXML+XBRL Server con Altova LicenseServer y asignarle licencias, consulte el apartado Asignación de licencias en Mac OS X.

Versiones de LicenseServer y compatibilidad con RaptorXML+XBRL Server
Cuando actualice RaptorXML+XBRL Server a una versión más reciente, asegúrese de actualizar

 © 2014 Altova GmbH

también su servidor LicenseServer con la versión más reciente. Las versiones anteriores de RaptorXML+XBRL Server funcionarán con las conversiones más recientes de LicenseServer.

Es decir, si instala una versión nueva de RaptorXML+XBRL Server y la versión actual de su servidor LicenseServer no es la más reciente, desinstale la versión antigua de LicenseServer e instale la versión más reciente (disponible en el sitio web de Altova). Toda la información de registro y licencias almacenada en la versión antigua se guardará en una base de datos su equipo servidor durante la desinstalación y se importará automáticamente a la versión nueva.

2.3.2 Asignación de licencias en Mac OS X

Temas de este apartado:

- Cómo asignar licencias
- Nota sobre el usuario raíz
- Iniciar LicenseServer como servicio
- Iniciar RaptorXML+XBRL Server como servicio
- Registrar RaptorXML+XBRL Server
- Nota sobre núcleos y licencias

Cómo asignar licencias
Siga estos pasos para asignar una licencia a RaptorXML+XBRL Server en sistemas Mac OS X con un servidor Altova LicenseServer instalado en un equipo de la red:

1. Si Altova LicenseServer todavía no está ejecutándose, inicie Altova LicenseServer como servicio.
2. Registre RaptorXML+XBRL Server con LicenseServer.
3. En la pestaña **Server Management** de la página de configuración de LicenseServer, asigne una licencia a RaptorXML+XBRL Server.

Nota sobre el usuario raíz
Para registrar RaptorXML+XBRL Server con LicenseServer en Mac OS X es necesario tener privilegios de administrador (usuario raíz "root"). Por tanto, debe realizar la asignación de licencias como usuario raíz.

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

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

Si por cualquier motivo necesita detener LicenseServer, use el mismo comando pero sustituya `load` por `unload`. Por ejemplo:

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

Iniciar RaptorXML+XBRL Server como servicio
Use este comando para iniciar RaptorXML+XBRL Server como servicio:

```
sudo launchctl load /Library/LaunchDaemons/
```

 © 2014 Altova GmbH

```
com.altova.RaptorXMLXBRLServer2014.plist
```

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

```
sudo launchctl unload /Library/LaunchDaemons/
com.altova.RaptorXMLXBRLServer2014.plist
```

Registrar RaptorXML+XBRL Server

Para poder asignarle una licencia, antes debe registrar RaptorXML+XBRL Server con LicenseServer. Para ello puede utilizar el comando `licenseserver` de la interfaz de la línea de comandos de RaptorXML+XBRL Server. Por ejemplo:

```
sudo /usr/local/Altova/RaptorXMLXBRLServer2014/bin/RaptorXMLXBRL licenseserver
localhost
```

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

```
/usr/local/Altova/RaptorXMLXBRLServer2014/bin
```

Tras registrarse, RaptorXML+XBRL Server aparece en la lista de la pestaña **Server Management** de la página de configuración de LicenseServer, donde puede asignarle una licencia.

Nota sobre núcleos y licencias

La asignación de licencias para RaptorXML+XBRL Server depende del número de núcleos disponibles en el equipo donde se ejecuta RaptorXML+XBRL 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.

 © 2014 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.

 © 2014 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\RaptorXMLXBRLServer2014\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\RaptorXMLXBRLServer2014\etc`. Cada archivo `catalog.xml` está en una carpeta de esquema y estas carpetas de esquema están 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

 © 2014 Altova GmbH

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.

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\Common2014`
`%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.

 © 2014 Altova GmbH

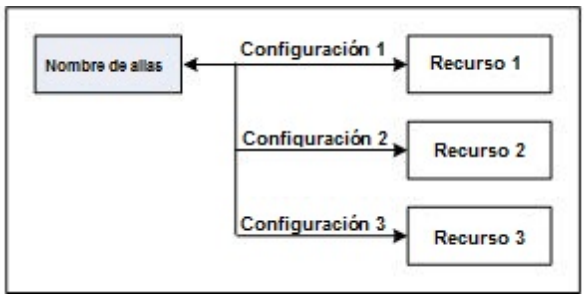
`%LocalAppDataFolder%`	Ruta de acceso completa al directorio del sistema de archivos que sirve como repositorio de datos para aplicaciones locales (no roaming).
`%MyPicturesFolder%`	Ruta de acceso completa a la carpeta `Mis imágenes`.

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:

 © 2014 Altova GmbH

```
raptorxmlxbrl 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:

```
raptorxmlxbrl 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:

```
raptorxmlxbrl 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`**.

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+XBRL 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 instale y configure RaptorXML +XBRL 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+XBRL 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.

 © 2014 Altova GmbH

Altova RaptorXML+XBRL Server 2014

Interfaz de la línea de comandos (ILC)

3 Interfaz de la línea de comandos (ILC)

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

Windows `<CarpetaArchivosDePrograma>\Altova\RaptorXMLXBRLServer2014\bin\`**`RaptorXMLXBRL.exe`**
Linux `\opt\Altova\RaptorXMLXBRLServer2014\bin\`**`raptorxmlxbrl`**
Mac `\usr\local\Altova\RaptorXMLXBRLServer2014\bin\`**`raptorxmlxbrl`**

Uso de mayúsculas/minúsculas en la línea de comandos

`RaptorXMLXBRL` *en Windows*
`raptorxmlxbrl` *en Unix (Linux, Mac)*

* Nota: las minúsculas (`raptorxmlxbrl`) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que la mezcla de mayúsculas y minúsculas (`RaptorXMLXBRL`) solo funciona en Windows y Mac.

Uso
Esta es la sintaxis de la línea de comandos:

Windows **`RaptorXMLXBRL --h | --help | --version | <comando> [opciones] [argumentos]`**
Linux **`raptorxmlxbrl --h | --help | --version | <comando> [opciones] [argumentos]`**
Mac **`raptorxmlxbrl --h | --help | --version | <comando> [opciones] [argumentos]`**

`RaptorXMLXBRL`	Llama a la aplicación (en plataformas Windows)
`raptorxmlxbrl`	Llama a la aplicación (en plataformas Unix: Linux y Mac)
`--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.

 © 2014 Altova GmbH

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.

Todos los 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.
`valxbrl | xbrl`	Valida una instancia XBRL (`*.xbrl`).
`valxbrltaxonomy | dts`	Valida una taxonomía (esquema) XBRL (`*.xsd`).
`valxquery`	Valida un documento XQuery.
`valxsd | xsd`	Valida un documento de 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.

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 de validación XBRL

`valxbrl | xbrl`	Valida una instancia XBRL (`*.xbrl`).
`valxbrltaxonomy | dts`	Valida una taxonomía (esquema) XBRL (`*.xsd`).
`valany`	Valida cualquier tipo de documento de instancia o taxonomía XBRL. El tipo de documento se detecta de forma automática.

Comandos XSLT

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

Comandos XQuery

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

© 2014 Altova GmbH

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.

3.1.1 valxml-withdtd (xml)

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

Windows	**RaptorXMLXBRL valxml-withdtd | xml [opciones] *ArchivoEntrada***
Linux	**raptorxmlxbrl valxml-withdtd | xml [opciones] *ArchivoEntrada***
Mac	**raptorxmlxbrl 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

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

Uso de mayúsculas/minúsculas en la línea de comandos

RaptorXMLXBRL *en Windows*
raptorxmlxbrl *en Unix (Linux, Mac)*

* Nota: las minúsculas (`raptorxmlxbrl`) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que la mezcla de mayúsculas y minúsculas (`RaptorXMLXBRL`) solo funciona en Windows y Mac.

Opciones
Las opciones del comando aparecen más abajo divididas en grupos. Los valores se pueden dar sin comillas excepto en estos dos casos: (i) cuando la cadena de valor contiene espacios y (ii) cuando en la descripción de la opción se indique explícitamente que es necesario el uso de comillas.

Validación y procesamiento

dtd

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

 © 2014 Altova GmbH

listfile

--listfile = `true|false`

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

namespaces

--namespaces = `true|false`

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

recurse

--recurse = `true|false`

Esta opción se utiliza para seleccionar ficheros dentro de 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`. Valor predeterminado: `false`
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

streaming

--streaming = `true|false`

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

Catálogos y recursos globales

catalog

--catalog = *`ARCHIVO`*

Especifica la ruta de acceso absoluta a un archivo de catálogo que no está en el archivo de catálogo raíz instalado. El valor predeterminado es la ruta de acceso absoluta del archivo de catálogo raíz instalado (`<carpeta-instalación>\Altova\RaptorXMLXBRLServer2014\etc\RootCatalog.xml`). Consulte el apartado Catálogos XML para obtener más información.

user-catalog

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

enable-globalresources

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

globalresourceconfig [gc]

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

globalresourcefile [gr]

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

Mensajes, errores y ayuda

error-format

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

error-limit

`--error-limit =` *`N`*
Especifica el límite de errores. Se permiten los valores comprendidos entre el `1` y el `999`. 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`.

help

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

log-output

`--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.

 © 2014 Altova GmbH

verbose

--verbose = `true|false`

Si el valor es `true`, se genera información adicional durante la validación. Valor predeterminado es `false`.

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

version

--version

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

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.

Windows	**`RaptorXMLXBRL valxml-withxsd	xsi [opciones] ArchivoEntrada`**
Linux	**`raptorxmlxbrl valxml-withxsd	xsi [opciones] ArchivoEntrada`**
Mac	**`raptorxmlxbrl 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

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

▼ Uso de mayúsculas/minúsculas en la línea de comandos

`RaptorXMLXBRL` *en Windows*
`raptorxmlxbrl` *en Unix (Linux, Mac)*

* Nota: las minúsculas (`raptorxmlxbrl`) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que la mezcla de mayúsculas y minúsculas (`RaptorXMLXBRL`) solo funciona en Windows y Mac.

Opciones

Las opciones del comando aparecen más abajo divididas en grupos. Los valores se pueden dar sin comillas excepto en estos dos casos: (i) cuando la cadena de valor contiene espacios y (ii) cuando en la descripción de la opción se indique explícitamente que es necesario el uso de comillas.

▼ Validación y procesamiento

▼ **`assessment-mode`**

 © 2014 Altova GmbH

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

listfile

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

parallel-assessment [pa]

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

recurse

--recurse = true|false
Esta opción se utiliza para seleccionar ficheros dentro de 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. Valor predeterminado: false
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en true.

schema-imports

--schema-imports = load-by-schemalocation | load-preferring-schemalocation | load-by-namespace | load-combining-both | license-namespace-only
Esta opción indica el comportamiento de los elementos xs:import. Cada uno de estos elementos tiene un atributo opcional namespace y un atributo opcional schemaLocation: <import 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**.
- `load-by-namespace`: el valor del atributo `namespace` se utiliza para buscar el esquema por medio de una asignación de catálogo.
- `load-combining-both`: si el atributo `namespace` o `schemaLocation` tiene una asignación de catálogo, entonces se usa la asignación. Si ambos atributos tienen asignaciones de catálogo, entonces es el valor de la opción `--schema-mapping` (opción XBRL y opción XML/XSD) decide qué asignación se utiliza. Si no hay ninguna asignación de catálogo, entonces se usa el atributo `schemaLocation`.
- `license-namespace-only`: se importa el espacio de nombres. No se importa el documento de esquema.

schema-location-hints

`--schema-location-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 de instancia XML o XBRL.
- 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` (opción XBRL y opción XML/XSD) decide qué asignación se utiliza. Si ni el espacio de nombres ni la URL tiene una asignación de catálogo, se usa la URL.
- El valor `ignore` ignora los atributos `xsi:schemaLocation` y `xsi:noNamespaceSchemaLocation`.

schema-mapping

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

Si 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). Valor predeterminado: `prefer-schemalocation`.

script

`--script =` *`ARCHIVO`*

Una vez finalizada la validación, ejecuta el script Python.

 © 2014 Altova GmbH

streaming

--streaming = `true|false`

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

xinclude

--xinclude = `true|false`

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

xml-mode

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

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

xsd

--xsd = *ARCHIVO*

Especifica qué esquemas XML deben utilizarse para la validación de documentos XML. Si quiere especificar más de un esquema, añada la opción varias veces.

xsd-version

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

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

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

Catálogos y recursos globales

catalog

--catalog = *ARCHIVO*
Especifica la ruta de acceso absoluta a un archivo de catálogo que no está en el archivo de catálogo raíz instalado. El valor predeterminado es la ruta de acceso absoluta del archivo de catálogo raíz instalado (`<carpeta-instalación>\Altova\RaptorXMLXBRLServer2014\etc\RootCatalog.xml`). Consulte el apartado Catálogos XML para obtener más información.

user-catalog

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

enable-globalresources

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

globalresourceconfig [gc]

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

globalresourcefile [gr]

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

Mensajes, errores y ayuda

error-format

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

error-limit

--error-limit = *N*
Especifica el límite de errores. Se permiten los valores comprendidos entre el `1` y el `999`. 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`.

help

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

log-output

--log-output = *ARCHIVO*
Escribe el mensaje de salida en la URL de archivo indicada, en lugar de en la consola.

 © 2014 Altova GmbH

Compruebe que la ILC tiene permiso de escritura en la ubicación de destino.

verbose

--verbose = `true|false`

Si el valor es `true`, se genera información adicional durante la validación. Valor predeterminado es `false`.

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

version

--version

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

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.

Windows	**RaptorXMLXBRL valdtd | dtd [opciones] *ArchivoEntrada***
Linux	**raptorxmlxbrl valdtd | dtd [opciones] *ArchivoEntrada***
Mac	**raptorxmlxbrl 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

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

Uso de mayúsculas/minúsculas en la línea de comandos

RaptorXMLXBRL *en Windows*
raptorxmlxbrl *en Unix (Linux, Mac)*

* Nota: las minúsculas (`raptorxmlxbrl`) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que la mezcla de mayúsculas y minúsculas (`RaptorXMLXBRL`) solo funciona en Windows y Mac.

Opciones
Las opciones del comando aparecen más abajo divididas en grupos. Los valores se pueden dar sin comillas excepto en estos dos casos: (i) cuando la cadena de valor contiene espacios y (ii) cuando en la descripción de la opción se indique explícitamente que es necesario el uso de comillas.

Validación y procesamiento

listfile

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

 © 2014 Altova GmbH

- **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. Valor predeterminado: false
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en true.

- Catálogos y recursos globales
  - **catalog**

    --catalog = *ARCHIVO*
    Especifica la ruta de acceso absoluta a un archivo de catálogo que no está en el archivo de catálogo raíz instalado. El valor predeterminado es la ruta de acceso absoluta del archivo de catálogo raíz instalado (<carpeta-instalación>\Altova\RaptorXMLXBRLServer2014\etc\RootCatalog.xml). Consulte el apartado Catálogos XML para obtener más información.

  - **user-catalog**

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

  - **enable-globalresources**

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

  - **globalresourceconfig [gc]**

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

  - **globalresourcefile [gr]**

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

- Mensajes, errores y ayuda
  - **error-format**

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

error-limit

--error-limit = *N*

Especifica el límite de errores. Se permiten los valores comprendidos entre el `1` y el `999`. 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`.

help

--help

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

log-output

--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

--verbose = `true|false`

Si el valor es `true`, se genera información adicional durante la validación. Valor predeterminado es `false`.

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

version

--version

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

 © 2014 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.

Windows	**RaptorXMLXBRL valxsd | xsd [opciones] *ArchivoEntrada***
Linux	**raptorxmlxbrl valxsd | xsd [opciones] *ArchivoEntrada***
Mac	**raptorxmlxbrl 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

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

▼ Uso de mayúsculas/minúsculas en la línea de comandos

RaptorXMLXBRL *en Windows*
raptorxmlxbrl *en Unix (Linux, Mac)*

* Nota: las minúsculas (raptorxmlxbrl) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que la mezcla de mayúsculas y minúsculas (RaptorXMLXBRL) solo funciona en Windows y Mac.

Opciones
Las opciones del comando aparecen más abajo divididas en grupos. Los valores se pueden dar sin comillas excepto en estos dos casos: (i) cuando la cadena de valor contiene espacios y (ii) cuando en la descripción de la opción se indique explícitamente que es necesario el uso de comillas.

▼ Validación y procesamiento

▼ **listfile**

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

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

recurse

`--recurse =` `true|false`
Esta opción se utiliza para seleccionar ficheros dentro de 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`. Valor predeterminado: `false`
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

schema-imports

`--schema-imports =` `load-by-schemalocation | load-preferring-schemalocation | load-by-namespace | load-combining-both | license-namespace-only`
Esta opción indica el comportamiento de los elementos `xs:import`. Cada uno de estos elementos tiene un atributo opcional `namespace` y un atributo opcional `schemaLocation`: `<import 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**.
- `load-by-namespace`: el valor del atributo `namespace` se utiliza para buscar el esquema por medio de una asignación de catálogo.
- `load-combining-both`: si el atributo `namespace` o `schemaLocation` tiene una asignación de catálogo, entonces se usa la asignación. Si ambos atributos tienen asignaciones de catálogo, entonces es el valor de la opción `--schema-mapping` (opción XBRL y opción XML/XSD) decide qué asignación se utiliza. Si no hay ninguna asignación de catálogo, entonces se usa el atributo `schemaLocation`.
- `license-namespace-only`: se importa el espacio de nombres. No se importa el documento de esquema.

schema-location-hints

`--schema-location-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 de instancia XML o XBRL.
- 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` (opción XBRL y opción XML/XSD) decide qué asignación se utiliza. Si ni el espacio de nombres ni la URL tiene una asignación de catálogo, se usa la URL.
- El valor `ignore` ignora los atributos `xsi:schemaLocation` y `xsi:noNamespaceSchemaLocation`.

schema-mapping

--schema-mapping = `prefer-schemalocation | prefer-namespace`
Si 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). Valor predeterminado: `prefer-schemalocation`.

script

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

xinclude

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

xml-mode

--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`.

xsd-version

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

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

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

entiende que la versión del esquema es `1.0`.

- Catálogos y recursos globales
  - **catalog**

    --catalog = *ARCHIVO*
    Especifica la ruta de acceso absoluta a un archivo de catálogo que no está en el archivo de catálogo raíz instalado. El valor predeterminado es la ruta de acceso absoluta del archivo de catálogo raíz instalado (`<carpeta-instalación>\Altova\RaptorXMLXBRLServer2014\etc\RootCatalog.xml`). Consulte el apartado Catálogos XML para obtener más información.

  - **user-catalog**

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

  - **enable-globalresources**

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

  - **globalresourceconfig [gc]**

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

  - **globalresourcefile [gr]**

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

- Mensajes, errores y ayuda
  - **error-format**

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

  - **error-limit**

    --error-limit = *N*
    Especifica el límite de errores. Se permiten los valores comprendidos entre el `1` y el `999`. 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`.

  - **help**

    --help
    Muestra el texto de ayuda para el comando. Por ejemplo `valany --h`. (Otra opción es

usar el comando `help` con un argumento. Por ejemplo: `help valany`).

log-output

--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

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

version

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

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.

Windows	**RaptorXMLXBRL valany [opciones] *ArchivoEntrada***
Linux	**raptorxmlxbrl valany [opciones] *ArchivoEntrada***
Mac	**raptorxmlxbrl 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

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

Uso de mayúsculas/minúsculas en la línea de comandos

RaptorXMLXBRL *en Windows*
raptorxmlxbrl *en Unix (Linux, Mac)*

* Nota: las minúsculas (raptorxmlxbrl) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que la mezcla de mayúsculas y minúsculas (RaptorXMLXBRL) solo funciona en Windows y Mac.

Opciones
Las opciones del comando aparecen más abajo divididas en grupos. Los valores se pueden dar sin comillas excepto en estos dos casos: (i) cuando la cadena de valor contiene espacios y (ii) cuando en la descripción de la opción se indique explícitamente que es necesario el uso de comillas.

Validación y procesamiento

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

schema-imports

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

`| load-by-namespace | load-combining-both | license-namespace-only`
Esta opción indica el comportamiento de los elementos `xs:import`. Cada uno de estos elementos tiene un atributo opcional `namespace` y un atributo opcional `schemaLocation`: `<import 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**.
- `load-by-namespace`: el valor del atributo `namespace` se utiliza para buscar el esquema por medio de una asignación de catálogo.
- `load-combining-both`: si el atributo `namespace` o `schemaLocation` tiene una asignación de catálogo, entonces se usa la asignación. Si ambos atributos tienen asignaciones de catálogo, entonces es el valor de la opción `--schema-mapping` (opción XBRL y opción XML/XSD) decide qué asignación se utiliza. Si no hay ninguna asignación de catálogo, entonces se usa el atributo `schemaLocation`.
- `license-namespace-only`: se importa el espacio de nombres. No se importa el documento de esquema.

schema-location-hints

`--schema-location-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 de instancia XML o XBRL.
- 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` (opción XBRL y opción XML/XSD) decide qué asignación se utiliza. Si ni el espacio de nombres ni la URL tiene una asignación de catálogo, se usa la URL.
- El valor `ignore` ignora los atributos `xsi:schemaLocation` y `xsi:noNamespaceSchemaLocation`.

schema-mapping

`--schema-mapping =` `prefer-schemalocation | prefer-namespace`
Si 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). Valor predeterminado: `prefer-schemalocation`.

- Catálogos y recursos globales
  - **catalog**

    --catalog = *ARCHIVO*
    Especifica la ruta de acceso absoluta a un archivo de catálogo que no está en el archivo de catálogo raíz instalado. El valor predeterminado es la ruta de acceso absoluta del archivo de catálogo raíz instalado (`<carpeta-instalación>\Altova\RaptorXMLXBRLServer2014\etc\RootCatalog.xml`). Consulte el apartado Catálogos XML para obtener más información.

  - **user-catalog**

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

  - **enable-globalresources**

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

  - **globalresourceconfig [gc]**

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

  - **globalresourcefile [gr]**

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

- Mensajes, errores y ayuda
  - **error-format**

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

  - **error-limit**

    --error-limit = *N*
    Especifica el límite de errores. Se permiten los valores comprendidos entre el `1` y el `999`. 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`.

  - **help**

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

 © 2014 Altova GmbH

log-output

--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

--verbose = `true|false`

Si el valor es `true`, se genera información adicional durante la validación. Valor predeterminado es `false`.

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

version

--version

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

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.

 © 2014 Altova GmbH

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.

Windows	**RaptorXMLXBRL wfxml [opciones] *ArchivoEntrada***
Linux	**raptorxmlxbrl wfxml [opciones] *ArchivoEntrada***
Mac	**raptorxmlxbrl 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

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

Uso de mayúsculas/minúsculas en la línea de comandos

RaptorXMLXBRL *en Windows*
raptorxmlxbrl *en Unix (Linux, Mac)*

* Nota: las minúsculas (`raptorxmlxbrl`) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que la mezcla de mayúsculas y minúsculas (`RaptorXMLXBRL`) solo funciona en Windows y Mac.

Opciones
Las opciones del comando aparecen más abajo divididas en grupos. Los valores se pueden dar sin comillas excepto en estos dos casos: (i) cuando la cadena de valor contiene espacios y (ii) cuando en la descripción de la opción se indique explícitamente que es necesario el uso de comillas.

Validación y procesamiento

dtd

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

listfile

--listfile = `true|false`

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

namespaces

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

recurse

`--recurse =` `true|false`
Esta opción se utiliza para seleccionar ficheros dentro de 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`. Valor predeterminado: `false`
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

Catálogos y recursos globales

catalog

`--catalog =` *`ARCHIVO`*
Especifica la ruta de acceso absoluta a un archivo de catálogo que no está en el archivo de catálogo raíz instalado. El valor predeterminado es la ruta de acceso absoluta del archivo de catálogo raíz instalado (`<carpeta-instalación>\Altova\RaptorXMLXBRLServer2014\etc\RootCatalog.xml`). Consulte el apartado Catálogos XML para obtener más información.

user-catalog

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

enable-globalresources

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

 © 2014 Altova GmbH

globalresourceconfig [gc]

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

globalresourcefile [gr]

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

Mensajes, errores y ayuda

error-format

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

error-limit

--error-limit = *N*
Especifica el límite de errores. Se permiten los valores comprendidos entre el `1` y el `999`. 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`.

help

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

log-output

--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

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

version

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

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.

Windows	**RaptorXMLXBRL wfdtd [opciones] *ArchivoEntrada***
Linux	**raptorxmlxbrl wfdtd [opciones] *ArchivoEntrada***
Mac	**raptorxmlxbrl 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

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

▼ Uso de mayúsculas/minúsculas en la línea de comandos

RaptorXMLXBRL *en Windows*
raptorxmlxbrl *en Unix (Linux, Mac)*

* Nota: las minúsculas (`raptorxmlxbrl`) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que la mezcla de mayúsculas y minúsculas (`RaptorXMLXBRL`) solo funciona en Windows y Mac.

Opciones

Las opciones del comando aparecen más abajo divididas en grupos. Los valores se pueden dar sin comillas excepto en estos dos casos: (i) cuando la cadena de valor contiene espacios y (ii) cuando en la descripción de la opción se indique explícitamente que es necesario el uso de comillas.

▼ Validación y procesamiento

▼ **listfile**

--listfile = `true|false`

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

 © 2014 Altova GmbH

- **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. Valor predeterminado: false
  Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en true.

- Catálogos y recursos globales
  - **catalog**

    --catalog = *ARCHIVO*
    Especifica la ruta de acceso absoluta a un archivo de catálogo que no está en el archivo de catálogo raíz instalado. El valor predeterminado es la ruta de acceso absoluta del archivo de catálogo raíz instalado (<carpeta-instalación>\Altova\RaptorXMLXBRLServer2014\etc\RootCatalog.xml). Consulte el apartado Catálogos XML para obtener más información.

  - **user-catalog**

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

  - **enable-globalresources**

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

  - **globalresourceconfig [gc]**

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

  - **globalresourcefile [gr]**

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

- Mensajes, errores y ayuda
  - **error-format**

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

error-limit

--error-limit = *N*

Especifica el límite de errores. Se permiten los valores comprendidos entre el `1` y el `999`. 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`.

help

--help

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

log-output

--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

--verbose = `true|false`

Si el valor es `true`, se genera información adicional durante la validación. Valor predeterminado es `false`.

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

version

--version

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

 © 2014 Altova GmbH

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.

Windows	**RaptorXMLXBRL wfany [opciones] *ArchivoEntrada***
Linux	**raptorxmlxbrl wfany [opciones] *ArchivoEntrada***
Mac	**raptorxmlxbrl 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

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

Uso de mayúsculas/minúsculas en la línea de comandos

RaptorXMLXBRL *en Windows*
raptorxmlxbrl *en Unix (Linux, Mac)*

* Nota: las minúsculas (raptorxmlxbrl) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que la mezcla de mayúsculas y minúsculas (RaptorXMLXBRL) solo funciona en Windows y Mac.

Opciones
Las opciones del comando aparecen más abajo divididas en grupos. Los valores se pueden dar sin comillas excepto en estos dos casos: (i) cuando la cadena de valor contiene espacios y (ii) cuando en la descripción de la opción se indique explícitamente que es necesario el uso de comillas.

Validación y procesamiento

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

Catálogos y recursos globales

catalog

--catalog = *ARCHIVO*
Especifica la ruta de acceso absoluta a un archivo de catálogo que no está en el archivo de catálogo raíz instalado. El valor predeterminado es la ruta de acceso absoluta del archivo de catálogo raíz instalado (`<carpeta-instalación>\Altova\RaptorXMLXBRLServer2014\etc\RootCatalog.xml`). Consulte el apartado Catálogos XML para obtener más información.

user-catalog

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

enable-globalresources

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

globalresourceconfig [gc]

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

globalresourcefile [gr]

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

Mensajes, errores y ayuda

error-format

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

error-limit

--error-limit = *N*
Especifica el límite de errores. Se permiten los valores comprendidos entre el `1` y el `999`. 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`.

help

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

log-output

--log-output = *ARCHIVO*
Escribe el mensaje de salida en la URL de archivo indicada, en lugar de en la consola.

 © 2014 Altova GmbH

Compruebe que la ILC tiene permiso de escritura en la ubicación de destino.

verbose

--verbose = `true|false`

Si el valor es `true`, se genera información adicional durante la validación. Valor predeterminado es `false`.

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

version

--version

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

3.3 Comandos de validación XBRL

Los comandos de validación XBRL sirven para validar documentos de instancia y taxonomías XBRL con las especificaciones XBRL 2.1, Dimensions 1.0 y Formula 1.0.

valxbrl | xbrl	Valida un documento de instancia XBRL (`*.xbrl`)
valxbrltaxonomy | dts	Valida una taxonomía (esquema) XBRL (`*.xsd`)
valany	Valida cualquier tipo de documento XBRL. Este comando también sirve para validar documentos XML, DTD, XSD, XSLT y XQuery. El tipo de documento se detecta automáticamente.

 © 2014 Altova GmbH

3.3.1 valxbrl (xbrl)

El comando **valxbrl | xbrl** valida uno o varios documentos de instancia XBRL con las especificaciones XBRL 2.1, Dimensions 1.0 y Formula 1.0.

Windows	**RaptorXMLXBRL valxbrl | xbrl [opciones] *ArchivoEntrada***
Linux	**raptorxmlxbrl valxbrl | xbrl [opciones] *ArchivoEntrada***
Mac	**raptorxmlxbrl valxbrl | xbrl [opciones] *ArchivoEntrada***

El argumento *ArchivoEntrada* es el documento de instancia XBRL 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*).

Nota: el documento de instancia XBRL no puede estar anidado en otro documento XML y su elemento raíz debe ser el elemento xbrl.

```
<xbrl xmlns="http://www.xbrl.org/2003/instance"> ... </xbrl>
```

Ejemplos

- **raptorxmlxbrl** valxbrl c:\Test.xbrl
- **raptorxmlxbrl** xbrl --formula-execution=true --formula-output=c:\FormulaSalida.xml c:\Test.xbrl
- **raptorxmlxbrl** xbrl --formula-execution --assertions-output=c:\AssertionsSalida.xml c:\Test.xbrl
- **raptorxmlxbrl** xbrl --formula-execution --formula-output=c:\FormulaSalida.xml --assertions-output=c:\AssertionsSalida.xml c:\Test.xbrl

Uso de mayúsculas/minúsculas en la línea de comandos

RaptorXMLXBRL *en Windows*
raptorxmlxbrl *en Unix (Linux, Mac)*

* Nota: las minúsculas (raptorxmlxbrl) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que la mezcla de mayúsculas y minúsculas (RaptorXMLXBRL) solo funciona en Windows y Mac.

Opciones
Las opciones del comando aparecen más abajo divididas en grupos. Los valores se pueden dar sin comillas excepto en estos dos casos: (i) cuando la cadena de valor contiene espacios y (ii) cuando en la descripción de la opción se indique explícitamente que es necesario el uso de comillas.

Validación y procesamiento XBRL

dimensions

--dimensions = true|false
Habilita las extensiones XBRL Dimension 1.0. Valor predeterminado: true.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en true.

listfile

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

parallel-assessment [pa]

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

preload-xbrl-schemas

--preload-xbrl-schemas = true|false
Carga previa de los esquemas de la especificación XBRL 2.1. Valor predeterminado: true.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en true.

recurse

--recurse = true|false
Esta opción se utiliza para seleccionar ficheros dentro de 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. Valor predeterminado: false
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en true.

 © 2014 Altova GmbH

schema-imports

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

Esta opción indica el comportamiento de los elementos `xs:import`. Cada uno de estos elementos tiene un atributo opcional `namespace` y un atributo opcional `schemaLocation`: `<import namespace="unEspacioNombres" schemaLocation="unaURL">`. 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**.
- `load-by-namespace`: el valor del atributo `namespace` se utiliza para buscar el esquema por medio de una asignación de catálogo.
- `load-combining-both`: si el atributo `namespace` o `schemaLocation` tiene una asignación de catálogo, entonces se usa la asignación. Si ambos atributos tienen asignaciones de catálogo, entonces es el valor de la opción `--schema-mapping` (opción XBRL y opción XML/XSD) decide qué asignación se utiliza. Si no hay ninguna asignación de catálogo, entonces se usa el atributo `schemaLocation`.
- `license-namespace-only`: se importa el espacio de nombres. No se importa el documento de esquema.

schema-location-hints

--schema-location-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 de instancia XML o XBRL.
- 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` (opción XBRL y opción XML/XSD) decide qué asignación se utiliza. Si ni el espacio de nombres ni la URL tiene una asignación de catálogo, se usa la URL.
- El valor `ignore` ignora los atributos `xsi:schemaLocation` y `xsi:noNamespaceSchemaLocation`.

schema-mapping

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

Si 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). Valor predeterminado: `prefer-schemalocation`.

script

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

treat-inconsistencies-as-errors

--treat-inconsistencies-as-errors = `true|false`
La validación XBRL falla si el archivo contiene incoherencias según lo definido por la especificación XBRL 2.1. El valor predeterminado es `false`.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

validate-dts-only

--validate-dts-only = `true|false`
La DTS se detecta iniciando desde el documento de instancia XBRL. Todos los esquemas de taxonomía y bases de enlace referenciados se detectan y validan. Los demás documentos de instancia XBRL se ignoran. Valor predeterminado: `false`.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

xinclude

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

Fórmulas y aserciones XBRL

assertions-output

--assertions-output = *ARCHIVO*
Escribe el resultado de la evaluación de la aserción en el *ARCHIVO* indicado. Si se define, especifica automáticamente `--formula-execution=true`.

assertions-output-format

--assertions-output-format = `json|xml`
Especifica el formato de salida de la evaluación de la aserción. Valor predeterminado: `json`.

formula

--formula = `true|false`
Habilita las extensiones XBRL Formula 1.0. Valor predeterminado: `true`.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

 © 2014 Altova GmbH

formula-assertion-set [[OBSOLETA]]

--formula-assertion-set = *VALOR*

Limita la ejecución de fórmulas al conjunto de aserciones dado. Añada la opción varias veces para especificar más de un grupo de aserciones. Su forma abreviada es `--as`. El *VALOR* es el valor del atributo `@id` o un URI con un fragmento XPointer que identifica el recurso. También puede usar los valores especiales `##none` y `##all`.

formula-execution

--formula-execution = `true|false`

Habilita la evaluación de fórmulas XBRL. Valor predeterminado: `true`. Si es `true`, especifica automáticamente `--formula=true`.

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

formula-output

--formula-output = *ARCHIVO*

Escribe el resultado de la evaluación de la fórmula en el *ARCHIVO* indicado. Si se define, especifica automáticamente `--formula-execution=true`.

formula-parameters [[OBSOLETA]]

--formula-parameters = *MATRIZ-JSON*

Especifica los parámetros para la evaluación de fórmulas XBRL en formato JSON en la interfaz de la línea de comandos directamente. Esta opción es **obsoleta** (utilice la opción `FormulaParametersFile`; véase Parámetros de fórmula).

El formato de la matriz JSON sería este:

```
  [{"name":"p1","namespace":"http://xbrl.org/2008/
conformance","type":"xs:string","value":"hello world from json"}, … ]
```

Sin embargo, tenga cuidado cuando use caracteres de escape en la línea de comandos. En el shell cmd.exe la opción `formula-parameters` tendría esta sintaxis:

```
--formula-parameters="[{\"name\":\"date\",\"namespace\":\"http://
example.com\",\"type\":\"xs:string\",\"value\":\"2007-12-31\"}]"
```

Puede que en otros sistemas operativos y shells sea necesario utilizar otros caracteres de escape.

formula-parameters-file

--formula-parameters-file = *ARCHIVO*

Especifica un *ARCHIVO* que contiene los parámetros para la evaluación de fórmulas XBRL. Puede tratarse de un archivo XML o de un archivo JSON. Consulte el apartado Parámetros de fórmula para obtener más información

- Formatos de parámetro antiguos (obsoletos)

  Formato de archivo XML:

```
<?xml version="1.0" encoding="utf-8"?>
  <parameters>
    <parameter name="p1" namespace="http://xbrl.org/2008/conformance"
type="xs:string" value="hello world from xml"/>
    ...
  </parameters>
```

  Formato JSON:

```
[{"name":"p1","namespace":"http://xbrl.org/2008/
conformance","type":"xs:string","value":"hello world from json"}, … ]
```

La clave `namespace` es opcional.

preload-formula-schemas

`--preload-formula-schemas =` `true|false`

Carga previa de los esquemas de la especificación XBRL Formula 1.0. Valor predeterminado: `false`.

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

process-assertion [a]

`--a | --process-assertion =` *VALOR*

Limita la ejecución de fórmulas a la aserción dada. Añada la opción varias veces para especificar más de una aserción. Su forma abreviada es `--a`. El *VALOR* es el valor del atributo `@id` o un URI con un fragmento XPointer que identifica el recurso. También se pueden usar los valores especiales `##none` y `##all`.

process-assertion-set [as]

`--as | --process-assertion-set =` *VALOR*

Limita la ejecución de fórmulas al conjunto de aserciones dado. Añada la opción varias veces para especificar más de un conjunto de aserciones. Su forma abreviada es `--as`. El ***VALOR*** es el valor del atributo `@id` o un URI con un fragmento XPointer que identifica el recurso. También puede usar los valores especiales `##none` y `##all`.

process-formula [f]

`--f | --process-formula =` *VALOR*

Limita la ejecución de fórmulas a la fórmula dada. Añada la opción varias veces para especificar más de una fórmula. Su forma abreviada es `--as`. El ***VALOR*** es el valor del atributo `@id` o un URI con un fragmento XPointer que identifica el recurso. También puede usar los valores especiales `##none` y `##all`.

variableset-execution-timeout

`--variableset-execution-timeout =` *VALOR*

Se aplica cuando se ejecutan fórmulas (**`--formula-execution=true`**). Especifica el tiempo de espera máximo permitido para ejecutar un conjunto de variables único (una fórmula o un valor o una aserción de existencia o de coherencia). El tiempo de espera se indica en minutos y debe ser un número positivo. El valor predeterminado es 30 minutos. Si un conjunto de variables determinado no termina de ejecutarse antes de cumplirse el tiempo de espera, la ejecución se anula. Aparece un mensaje de error (que se introduce en el registro detallado). No obstante, debe tener en cuenta que la comprobación de tiempo de espera solamente se realiza después de evaluarse cada conjunto de variables (y no durante la ejecución de las expresiones XPath). Por tanto, si una expresión XPath tarda mucho en ejecutarse, puede que se cumpla el tiempo de espera. La ejecución del conjunto de variables se anula solamente cuando se ejecute la evaluación completa del conjunto de variables.

Tablas XBRL

 © 2014 Altova GmbH

Propuesta de recomendación (no es recomendación final)
La implementación actual de RaptorXML+XBRL Server (publicada en marzo de 2014) sigue la propuesta de recomendación Table Linkbase del 18 de diciembre de 2013 y utiliza el espacio de nombres `http://xbrl.org/PR/2013-12-18/table`. Si antes de que Altova publique una versión nueva de RaptorXML+XBRL Server aparece una recomendación nueva o una especificación final, tenga en cuenta que el espacio de nombres que utiliza RaptorXML+XBRL Server será obsoleto y no cumplirá con la nueva recomendación/especificación. La implementación se actualizará en la versión de RaptorXML+XBRL Server inmediatamente posterior a la publicación de la nueva recomendación/especificación.

preload-table-schemas

--preload-table-schemas = `true|false`

Carga previa de los esquemas de la especificación XBRL Table 1.0. Valor predeterminado: `false`.

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

process-table [t]

--t | --process-table = *VALOR*

Limita la ejecución de fórmulas a la taba dada. Añada la opción varias veces para especificar más de una tabla. Su forma abreviada es `--t`. El ***VALOR*** es el valor del atributo `@id` o un URI con un fragmento XPointer que identifica el recurso. También puede usar los valores especiales `##none` y `##all`.

table

--table = `true|false`

Habilita la extensión XBRL Table 1.0. Si es `true`, especifica automáticamente `--formula=true` y `--dimensions=true`. Valor predeterminado: `true`.

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

table-elimination

--table-elimination = `true|false`

Habilita la eliminación de columnas/filas de tabla vacías en el formato de salida HTML. Valor predeterminado: `true`.

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

table-execution

--table-execution = `true|false`

Habilita la evaluación de tablas XBRL. Valor predeterminado: `false`. Si se especifica la opción `--table-output`, a esta opción se asigna `true`. Si tiene el valor `true`, se especifica automáticamente `--table=true`.

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

table-output

--table-output = *ARCHIVO*

Escribe la tabla de salida en el ***ARCHIVO*** indicado. Si se define, especifica automáticamente `--table-execution=true`.

table-output-format

--table-output-format = `xml|html`
Especifica el formato de la tabla de salida. Valor predeterminado: `xml`.

Etiquetas XBRL

concept-label-linkrole

--concept-label-linkrole = *VALOR*
Especifica la función de vínculo preferida a la hora de representar etiquetas de concepto.

concept-label-role

--cconcept-label-role = *VALOR*
Especifica la función de etiqueta preferida a la hora de representar etiquetas de concepto. La función predeterminada es `http://www.xbrl.org/2003/role/label`.

generic-label-linkrole

--generic-label-linkrole = *VALOR*
Especifica la función de vínculo extendido preferida a la hora de representar etiquetas genéricas.

generic-label-role

--generic-label-role = *VALOR*
Especifica la función de etiqueta preferida a la hora de representar etiquetas genéricas. La función predeterminada es `http://www.xbrl.org/2003/role/label`

label-lang

--label-lang = *VALOR*
Especifica el idioma de etiqueta preferido a la hora de representar etiquetas. Valor predeterminado: `en`

Catálogos y recursos globales

catalog

--catalog = *ARCHIVO*
Especifica la ruta de acceso absoluta a un archivo de catálogo que no está en el archivo de catálogo raíz instalado. El valor predeterminado es la ruta de acceso absoluta del archivo de catálogo raíz instalado (`<carpeta-instalación>\Altova\RaptorXMLXBRLServer2014\etc\RootCatalog.xml`). Consulte el apartado Catálogos XML para obtener más información.

user-catalog

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

 © 2014 Altova GmbH

enable-globalresources

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

globalresourceconfig [gc]

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

globalresourcefile [gr]

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

3.3.2 valxbrltaxonomy (dts)

El comando **valxbrltaxonomy | dts** valida una o varias taxonomías (esquemas) XBRL con las especificaciones XBRL 2.1, Dimensions 1.0 y Formula 1.0.

Windows	**RaptorXMLXBRL valxbrltaxonomy | dts [opciones] *ArchivoEntrada***
Linux	**raptorxmlxbrl valxbrltaxonomy | dts [opciones] *ArchivoEntrada***
Mac	**raptorxmlxbrl valxbrltaxonomy | dts [opciones] *ArchivoEntrada***

El argumento *ArchivoEntrada* es la taxonomía XBRL 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

- **raptorxmlxbrl** `valxbrltaxonomy c:\Test.xsd`
- **raptorxmlxbrl** `dts --listfile c:\ListaArchivos.txt`

▼ Uso de mayúsculas/minúsculas en la línea de comandos

RaptorXMLXBRL *en Windows*
raptorxmlxbrl *en Unix (Linux, Mac)*

* Nota: las minúsculas (`raptorxmlxbrl`) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que la mezcla de mayúsculas y minúsculas (`RaptorXMLXBRL`) solo funciona en Windows y Mac.

Opciones

Las opciones del comando aparecen más abajo divididas en grupos. Los valores se pueden dar sin comillas excepto en estos dos casos: (i) cuando la cadena de valor contiene espacios y (ii) cuando en la descripción de la opción se indique explícitamente que es necesario el uso de comillas.

▼ Validación y procesamiento XBRL

▼ **dimensions**

--dimensions = `true|false`
Habilita las extensiones XBRL Dimension 1.0. Valor predeterminado: `true`.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

▼ **formula**

--formula = `true|false`
Habilita las extensiones XBRL Formula 1.0. Valor predeterminado: `true`.

 © 2014 Altova GmbH

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

listfile

`--listfile =` `true|false`

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

preload-formula-schemas

`--preload-formula-schemas =` `true|false`

Carga previa de los esquemas de la especificación XBRL Formula 1.0. Valor predeterminado: `false`.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

preload-xbrl-schemas

`--preload-xbrl-schemas =` `true|false`

Carga previa de los esquemas de la especificación XBRL 2.1. Valor predeterminado: `true`.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

recurse

`--recurse =` `true|false`

Esta opción se utiliza para seleccionar ficheros dentro de 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`. Valor predeterminado: `false`
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

schema-imports

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

Esta opción indica el comportamiento de los elementos `xs:import`. Cada uno de estos elementos tiene un atributo opcional `namespace` y un atributo opcional `schemaLocation`: `<import 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**.
- `load-by-namespace`: el valor del atributo `namespace` se utiliza para buscar el esquema por medio de una asignación de catálogo.
- `load-combining-both`: si el atributo `namespace` o `schemaLocation` tiene una asignación de catálogo, entonces se usa la asignación. Si ambos atributos tienen asignaciones de catálogo, entonces es el valor de la opción `--schema-mapping` (opción XBRL y opción XML/XSD) decide qué asignación se utiliza. Si no hay ninguna asignación de catálogo, entonces se usa el atributo `schemaLocation`.
- `license-namespace-only`: se importa el espacio de nombres. No se importa el documento de esquema.

schema-location-hints

--schema-location-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 de instancia XML o XBRL.
- 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` (opción XBRL y opción XML/XSD) decide qué asignación se utiliza. Si ni el espacio de nombres ni la URL tiene una asignación de catálogo, se usa la URL.
- El valor `ignore` ignora los atributos `xsi:schemaLocation` y `xsi:noNamespaceSchemaLocation`.

schema-mapping

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

Si 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). Valor predeterminado: `prefer-schemalocation`.

script

--script = *ARCHIVO*

Una vez finalizada la validación, ejecuta el script Python.

treat-inconsistencies-as-errors

--treat-inconsistencies-as-errors = `true|false`

 © 2014 Altova GmbH

La validación XBRL falla si el archivo contiene incoherencias según lo definido por la especificación XBRL 2.1. El valor predeterminado es `false`.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

xinclude

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

Tablas XBRL

Propuesta de recomendación (no es recomendación final)
La implementación actual de RaptorXML+XBRL Server (publicada en marzo de 2014) sigue la propuesta de recomendación Table Linkbase del 18 de diciembre de 2013 y utiliza el espacio de nombres `http://xbrl.org/PR/2013-12-18/table`. Si antes de que Altova publique una versión nueva de RaptorXML+XBRL Server aparece una recomendación nueva o una especificación final, tenga en cuenta que el espacio de nombres que utiliza RaptorXML+XBRL Server será obsoleto y no cumplirá con la nueva recomendación/especificación. La implementación se actualizará en la versión de RaptorXML+XBRL Server inmediatamente posterior a la publicación de la nueva recomendación/especificación.

preload-table-schemas

--preload-table-schemas = `true|false`
Carga previa de los esquemas de la especificación XBRL Table 1.0. Valor predeterminado: `false`.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

process-table [t]

--t | --process-table = *VALOR*
Limita la ejecución de fórmulas a la taba dada. Añada la opción varias veces para especificar más de una tabla. Su forma abreviada es `--t`. El ***VALOR*** es el valor del atributo `@id` o un URI con un fragmento XPointer que identifica el recurso. También puede usar los valores especiales `##none` y `##all`.

table

--table = `true|false`
Habilita la extensión XBRL Table 1.0. Si es `true`, especifica automáticamente `--formula=true` y `--dimensions=true`. Valor predeterminado: `true`.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

table-execution

--table-execution = `true|false`
Habilita la evaluación de tablas XBRL. Valor predeterminado: `false`. Si se especifica la opción `--table-output`, a esta opción se asigna `true`. Si tiene el valor `true`, se

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

table-output

`--table-output =` *`ARCHIVO`*
Escribe la tabla de salida en el ***ARCHIVO*** indicado. Si se define, especifica automáticamente `--table-execution=true`.

table-output-format

`--table-output-format =` `xml|html`
Especifica el formato de la tabla de salida. Valor predeterminado: `xml`.

Catálogos y recursos globales

catalog

`--catalog =` *`ARCHIVO`*
Especifica la ruta de acceso absoluta a un archivo de catálogo que no está en el archivo de catálogo raíz instalado. El valor predeterminado es la ruta de acceso absoluta del archivo de catálogo raíz instalado (`<carpeta-instalación>\Altova\RaptorXMLXBRLServer2014\etc\RootCatalog.xml`). Consulte el apartado Catálogos XML para obtener más información.

user-catalog

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

enable-globalresources

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

globalresourceconfig [gc]

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

globalresourcefile [gr]

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

Mensajes, errores y ayuda

error-format

`--error-format =` `text|shortxml|longxml`
Especifica el formato de la salida de error. Los valores posibles son formatos de texto,

 © 2014 Altova GmbH

XML y XML detallado (`longxml`). Valor predeterminado: `text`.

error-limit

`--error-limit =` *`N`*

Especifica el límite de errores. Se permiten los valores comprendidos entre el `1` y el `999`. 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`.

help

`--help`

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

log-output

`--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

`--verbose =` `true|false`

Si el valor es `true`, se genera información adicional durante la validación. Valor predeterminado es `false`.

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

version

`--version`

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

3.3.3 valany

El comando **valany** valida un documento de instancia o taxonomía XBRL con las especificaciones XBRL 2.1, Dimensions 1.0 y Formula 1.0 specifications. El tipo de documento se detecta automáticamente.

Windows	**RaptorXMLXBRL valany [opciones] *ArchivoEntrada***
Linux	**raptorxmlxbrl valany [opciones] *ArchivoEntrada***
Mac	**raptorxmlxbrl valany [opciones] *ArchivoEntrada***

El argumento *ArchivoEntrada* es 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

- **raptorxmlxbrl** valany c:\Test.xsd
- **raptorxmlxbrl** valany --errorformat=text c:\Test.xbrl

Uso de mayúsculas/minúsculas en la línea de comandos

RaptorXMLXBRL *en Windows*
raptorxmlxbrl *en Unix (Linux, Mac)*

* Nota: las minúsculas (raptorxmlxbrl) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que la mezcla de mayúsculas y minúsculas (RaptorXMLXBRL) solo funciona en Windows y Mac.

Opciones
Las opciones del comando aparecen más abajo divididas en grupos. Los valores se pueden dar sin comillas excepto en estos dos casos: (i) cuando la cadena de valor contiene espacios y (ii) cuando en la descripción de la opción se indique explícitamente que es necesario el uso de comillas.

Validación y procesamiento XBRL

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

schema-imports

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

Esta opción indica el comportamiento de los elementos `xs:import`. Cada uno de estos elementos tiene un atributo opcional `namespace` y un atributo opcional `schemaLocation`: `<import 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**.
- `load-by-namespace`: el valor del atributo `namespace` se utiliza para buscar el esquema por medio de una asignación de catálogo.
- `load-combining-both`: si el atributo `namespace` o `schemaLocation` tiene una asignación de catálogo, entonces se usa la asignación. Si ambos atributos tienen asignaciones de catálogo, entonces es el valor de la opción `--schema-mapping` (opción XBRL y opción XML/XSD) decide qué asignación se utiliza. Si no hay ninguna asignación de catálogo, entonces se usa el atributo `schemaLocation`.
- `license-namespace-only`: se importa el espacio de nombres. No se importa el documento de esquema.

schema-location-hints

--schema-location-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 de instancia XML o XBRL.
- 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` (opción XBRL y opción XML/XSD) decide qué asignación se utiliza. Si ni el espacio de nombres ni la URL tiene una asignación de catálogo, se usa la URL.
- El valor `ignore` ignora los atributos `xsi:schemaLocation` y `xsi:noNamespaceSchemaLocation`.

schema-mapping

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

Si 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). Valor predeterminado:

`prefer-schemalocation`.

- Catálogos y recursos globales
  - **catalog**

    --catalog = *ARCHIVO*
    Especifica la ruta de acceso absoluta a un archivo de catálogo que no está en el archivo de catálogo raíz instalado. El valor predeterminado es la ruta de acceso absoluta del archivo de catálogo raíz instalado (`<carpeta-instalación>\Altova\RaptorXMLXBRLServer2014\etc\RootCatalog.xml`). Consulte el apartado Catálogos XML para obtener más información.

  - **user-catalog**

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

  - **enable-globalresources**

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

  - **globalresourceconfig [gc]**

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

  - **globalresourcefile [gr]**

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

- Mensajes, errores y ayuda
  - **error-format**

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

  - **error-limit**

    --error-limit = *N*
    Especifica el límite de errores. Se permiten los valores comprendidos entre el `1` y el `999`. 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`.

  - **help**

    --help
    Muestra el texto de ayuda para el comando. Por ejemplo `valany --h`. (Otra opción es

usar el comando `help` con un argumento. Por ejemplo: `help valany`).

log-output

--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

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

version

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

3.4 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.

 © 2014 Altova GmbH

3.4.1 xslt

El comando **xslt** toma un archivo XSLT como único argumento y lo utiliza para transformar un archivo XML de entrada y generar un archivo de salida. Los archivos de entrada y salida se especifican como opciones.

Windows	**RaptorXMLXBRL xslt [opciones] *Archivo-XSLT***
Linux	**raptorxmlxbrl xslt [opciones] *Archivo-XSLT***
Mac	**raptorxmlxbrl 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

- **raptorxmlxbrl** xslt --input=c:\Test.xml --output=c:\Salida.xml c:\Test.xslt
- **raptorxmlxbrl** xslt --template-entry-point=PlantillaInicial --output=c:\Salida.xml c:\Test.xslt
- **raptorxmlxbrl** xslt --input=c:\Test.xml --output=c:\Salida.xml --param=date://node[1]/@att1 --p=title:'cadenasinespacios' --param=title:"'cadena con espacios'" --p=amount:456 c:\Test.xslt

▼ Uso de mayúsculas/minúsculas en la línea de comandos

RaptorXMLXBRL *en Windows*
raptorxmlxbrl *en Unix (Linux, Mac)*

* Nota: las minúsculas (raptorxmlxbrl) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que la mezcla de mayúsculas y minúsculas (RaptorXMLXBRL) solo funciona en Windows y Mac.

Opciones

Las opciones del comando aparecen más abajo divididas en grupos. Los valores se pueden dar sin comillas excepto en estos dos casos: (i) cuando la cadena de valor contiene espacios y (ii) cuando en la descripción de la opción se indique explícitamente que es necesario el uso de comillas.

▼ Procesamiento XSLT

▼ **chartext-disable**

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

dotnetext-disable

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

indent-characters

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

input

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

javaext-barcode-location

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

javaext-disable

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

output

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.

param [p]

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

XQuery

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

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

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

 © 2014 Altova GmbH

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

  ```
  raptorxmlxbrl xslt --input=c:\Test.xml --output=c:\Salida.xml --
  param=date://node[1]/@att1 --p=title:cadenasinespacios --
  param=title:"'cadena con espacios'" --p=amount:456 c:\Test.xslt
  ```

- **streaming**

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

- **template-entry-point**

  --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**

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

- **xslt-version**

  --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`

- Instancias y esquemas XML
  - **schema-imports**

    --schema-imports = `load-by-schemalocation | load-preferring-schemalocation | load-by-namespace | load-combining-both | license-namespace-only`
    Esta opción indica el comportamiento de los elementos `xs:import`. Cada uno de estos

elementos tiene un atributo opcional `namespace` y un atributo opcional `schemaLocation`: `<import namespace="unEspacioNombres" schemaLocation="unaURL">`. 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**.
- `load-by-namespace`: el valor del atributo `namespace` se utiliza para buscar el esquema por medio de una asignación de catálogo.
- `load-combining-both`: si el atributo `namespace` o `schemaLocation` tiene una asignación de catálogo, entonces se usa la asignación. Si ambos atributos tienen asignaciones de catálogo, entonces es el valor de la opción `--schema-mapping` (opción XBRL y opción XML/XSD) decide qué asignación se utiliza. Si no hay ninguna asignación de catálogo, entonces se usa el atributo `schemaLocation`.
- `license-namespace-only`: se importa el espacio de nombres. No se importa el documento de esquema.

schema-location-hints

`--schema-location-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 de instancia XML o XBRL.
- 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` (opción XBRL y opción XML/XSD) decide qué asignación se utiliza. Si ni el espacio de nombres ni la URL tiene una asignación de catálogo, se usa la URL.
- El valor `ignore` ignora los atributos `xsi:schemaLocation` y `xsi:noNamespaceSchemaLocation`.

schema-mapping

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

Si 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). Valor predeterminado: `prefer-schemalocation`.

xinclude

 © 2014 Altova GmbH

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

xml-mode

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

xsd-version

--xsd-version = 1.0|1.1|detect

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

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

Catálogos y recursos globales

catalog

--catalog = *ARCHIVO*
Especifica la ruta de acceso absoluta a un archivo de catálogo que no está en el archivo de catálogo raíz instalado. El valor predeterminado es la ruta de acceso absoluta del archivo de catálogo raíz instalado (`<carpeta-instalación>\Altova\RaptorXMLXBRLServer2014\etc\RootCatalog.xml`). Consulte el apartado Catálogos XML para obtener más información.

user-catalog

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

enable-globalresources

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

globalresourceconfig [gc]

--gc | --globalresourceconfig = *VALOR*

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

- **globalresourcefile [gr]**

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

- Mensajes, errores y ayuda
  - **error-limit**

    --error-limit = *N*
    Especifica el límite de errores. Se permiten los valores comprendidos entre el `1` y el `999`. 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`.

  - **help**

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

  - **verbose**

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

  - **version**

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

 © 2014 Altova GmbH

3.4.2 valxslt

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

Windows	**RaptorXMLXBRL valxslt [opciones]** ***Archivo-XSLT***
Linux	**raptorxmlxbrl valxslt [opciones]** ***Archivo-XSLT***
Mac	**raptorxmlxbrl 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

- **raptorxmlxbrl** valxslt c:\Test.xslt
- **raptorxmlxbrl** valxslt --xslt-version=2 c:\Test.xslt

Uso de mayúsculas/minúsculas en la línea de comandos

RaptorXMLXBRL *en Windows*
raptorxmlxbrl *en Unix (Linux, Mac)*

* Nota: las minúsculas (raptorxmlxbrl) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que la mezcla de mayúsculas y minúsculas (RaptorXMLXBRL) solo funciona en Windows y Mac.

Opciones
Las opciones del comando aparecen más abajo divididas en grupos. Los valores se pueden dar sin comillas excepto en estos dos casos: (i) cuando la cadena de valor contiene espacios y (ii) cuando en la descripción de la opción se indique explícitamente que es necesario el uso de comillas.

Procesamiento XSLT

chartext-disable

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

dotnetext-disable

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

javaext-barcode-location

--javaext-barcode-location = *ARCHIVO*

Especifica la ubicación del archivo de extensión de código de barras.

javaext-disable

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

template-entry-point

--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

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

xslt-version

--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`

Instancias y esquemas XML

schema-imports

--schema-imports = `load-by-schemalocation | load-preferring-schemalocation | load-by-namespace | load-combining-both | license-namespace-only`
Esta opción indica el comportamiento de los elementos `xs:import`. Cada uno de estos elementos tiene un atributo opcional `namespace` y un atributo opcional `schemaLocation`: `<import namespace="unEspacioNombres" schemaLocation="unaURL">`. 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**.
- `load-by-namespace`: el valor del atributo `namespace` se utiliza para buscar el esquema por medio de una asignación de catálogo.
- `load-combining-both`: si el atributo `namespace` o `schemaLocation` tiene una asignación de catálogo, entonces se usa la asignación. Si ambos atributos tienen asignaciones de catálogo, entonces es el valor de la opción `--schema-mapping` (opción XBRL y opción XML/XSD) decide qué asignación se utiliza. Si no hay ninguna asignación de catálogo, entonces se usa el atributo `schemaLocation`.
- `license-namespace-only`: se importa el espacio de nombres. No se importa el documento de esquema.

 © 2014 Altova GmbH

schema-location-hints

--schema-location-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 de instancia XML o XBRL.
- 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` (opción XBRL y opción XML/XSD) decide qué asignación se utiliza. Si ni el espacio de nombres ni la URL tiene una asignación de catálogo, se usa la URL.
- El valor `ignore` ignora los atributos `xsi:schemaLocation` y `xsi:noNamespaceSchemaLocation`.

schema-mapping

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

Si 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). Valor predeterminado: `prefer-schemalocation`.

xinclude

--xinclude = `true|false`

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

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

xml-mode

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

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

xsd-version

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

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

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

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

- Catálogos y recursos globales
  - **catalog**

    --catalog = *ARCHIVO*
    Especifica la ruta de acceso absoluta a un archivo de catálogo que no está en el archivo de catálogo raíz instalado. El valor predeterminado es la ruta de acceso absoluta del archivo de catálogo raíz instalado (`<carpeta-instalación>\Altova\RaptorXMLXBRLServer2014\etc\RootCatalog.xml`). Consulte el apartado Catálogos XML para obtener más información.

  - **user-catalog**

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

  - **enable-globalresources**

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

  - **globalresourceconfig [gc]**

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

  - **globalresourcefile [gr]**

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

- Mensajes, errores y ayuda
  - **error-limit**

    --error-limit = *N*
    Especifica el límite de errores. Se permiten los valores comprendidos entre el `1` y el `999`. 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`.

  - **help**

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

 © 2014 Altova GmbH

verbose

--verbose = `true|false`

Si el valor es `true`, se genera información adicional durante la validación. Valor predeterminado es `false`.

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

version

--version

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

3.5 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.

 © 2014 Altova GmbH

3.5.1 xquery

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

Windows	**RaptorXMLXBRL xquery [opciones] *Archivo-XQuery***
Linux	**raptorxmlxbrl xquery [opciones] *Archivo-XQuery***
Mac	**raptorxmlxbrl xquery [opciones] *Archivo-XQuery***

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

Ejemplos

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

Uso de mayúsculas/minúsculas en la línea de comandos

RaptorXMLXBRL *en Windows*
raptorxmlxbrl *en Unix (Linux, Mac)*

* Nota: las minúsculas (raptorxmlxbrl) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que la mezcla de mayúsculas y minúsculas (RaptorXMLXBRL) solo funciona en Windows y Mac.

Opciones
Las opciones del comando aparecen más abajo divididas en grupos. Los valores se pueden dar sin comillas excepto en estos dos casos: (i) cuando la cadena de valor contiene espacios y (ii) cuando en la descripción de la opción se indique explícitamente que es necesario el uso de comillas.

Procesamiento XQuery

indent-characters

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

input

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

omit-xml-declaration

--omit-xml-declaration = `true|false`

Opción de serialización que especifica si la declaración XML se omite en el resultado o no. Si el valor es `true`, el documento de salida no tendrá una declaración XML. Si el valor es `false`, se incluye una declaración XML en el documento de salida. Valor predeterminado: `false`.

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

output

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

--output-encoding = *VALOR*

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

output-indent

--output-indent = `true|false`

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

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

output-method

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

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

param [p]

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

XQuery

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

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

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

 © 2014 Altova GmbH

expresión XPath contiene espacios, entonces debe estar entre comillas dobles.

XSLT

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

```
raptorxmlxbrl xslt --input=c:\Test.xml --output=c:\Salida.xml --
param=date://node[1]/@att1 --p=title:cadenasinespacios --
param=title:"'cadena con espacios'" --p=amount:456 c:\Test.xslt
```

xquery-version

--xquery-version = `1|3`

Indica si el procesador XQuery debe usar XQuery 1.0 o 3.0. Valor predeterminado: `1`

Instancias y esquemas XML

xinclude

--xinclude = `true|false`

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

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

xml-mode

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

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

xsd-version

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

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

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

Catálogos y recursos globales

catalog

--catalog = *ARCHIVO*
Especifica la ruta de acceso absoluta a un archivo de catálogo que no está en el archivo de catálogo raíz instalado. El valor predeterminado es la ruta de acceso absoluta del archivo de catálogo raíz instalado (`<carpeta-instalación>\Altova\RaptorXMLXBRLServer2014\etc\RootCatalog.xml`). Consulte el apartado Catálogos XML para obtener más información.

user-catalog

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

enable-globalresources

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

globalresourceconfig [gc]

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

globalresourcefile [gr]

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

Mensajes, errores y ayuda

error-limit

--error-limit = *N*
Especifica el límite de errores. Se permiten los valores comprendidos entre el `1` y el `999`. 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`.

help

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

verbose

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

version

 © 2014 Altova GmbH

`--version`

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

3.5.2 valxquery

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

Windows	**RaptorXMLXBRL valxquery [opciones] *Archivo-XQuery***
Linux	**raptorxmlxbrl valxquery [opciones] *Archivo-XQuery***
Mac	**raptorxmlxbrl valxquery [opciones] *Archivo-XQuery***

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

Ejemplos

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

Uso de mayúsculas/minúsculas en la línea de comandos

RaptorXMLXBRL *en Windows*
raptorxmlxbrl *en Unix (Linux, Mac)*

* Nota: las minúsculas (raptorxmlxbrl) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que la mezcla de mayúsculas y minúsculas (RaptorXMLXBRL) solo funciona en Windows y Mac.

Opciones

Las opciones del comando aparecen más abajo divididas en grupos. Los valores se pueden dar sin comillas excepto en estos dos casos: (i) cuando la cadena de valor contiene espacios y (ii) cuando en la descripción de la opción se indique explícitamente que es necesario el uso de comillas.

Procesamiento XQuery

omit-xml-declaration

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

xquery-version

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

Instancias y esquemas XML

 © 2014 Altova GmbH

xinclude

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

xml-mode

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

xsd-version

--xsd-version = 1.0|1.1|detect

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

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

Catálogos y recursos globales

catalog

--catalog = *ARCHIVO*
Especifica la ruta de acceso absoluta a un archivo de catálogo que no está en el archivo de catálogo raíz instalado. El valor predeterminado es la ruta de acceso absoluta del archivo de catálogo raíz instalado (<carpeta-instalación>\Altova\RaptorXMLXBRLServer2014\etc\RootCatalog.xml). Consulte el apartado Catálogos XML para obtener más información.

user-catalog

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

enable-globalresources

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

globalresourceconfig [gc]

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

globalresourcefile [gr]

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

Mensajes, errores y ayuda

error-limit

--error-limit = *N*
Especifica el límite de errores. Se permiten los valores comprendidos entre el `1` y el `999`. 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`.

help

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

verbose

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

version

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

 © 2014 Altova GmbH

3.6 Comandos de ayuda y licencias

En esta sección describimos dos características importantes de RaptorXML+XBRL Server:

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

3.6.1 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.

Windows	**RaptorXMLXBRL help *Comando***
Linux	**raptorxmlxbrl help *Comando***
Mac	**raptorxmlxbrl 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

```
raptorxmlxbrl help valany
```

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

- Uso de mayúsculas/minúsculas en la línea de comandos

  RaptorXMLXBRL *en Windows*
  raptorxmlxbrl *en Unix (Linux, Mac)*

  * Nota: las minúsculas (`raptorxmlxbrl`) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que la mezcla de mayúsculas y minúsculas (`RaptorXMLXBRL`) solo funciona en Windows y Mac.

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`:

```
raptorxmlxbrl valany --help
```

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

```
raptorxmlxbrl help valany
```

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

 © 2014 Altova GmbH

3.6.2 Licencias

El comando `licenseserver` registra RaptorXML+XBRL Server 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.

Windows	**RaptorXMLXBRL licenseserver [opciones]** ***Servidor-O-Dirección-IP***
Linux	**raptorxmlxbrl licenseserver [opciones]** ***Servidor-O-Dirección-IP***
Mac	**raptorxmlxbrl licenseserver [opciones]** ***Servidor-O-Dirección-IP***

Tras registrar RaptorXML+XBRL Server 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+XBRL Server.

Ejemplo

```
raptorxmlxbrl licenseserver DOC.altova.com
```

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

▾ Uso de mayúsculas/minúsculas en la línea de comandos

`RaptorXMLXBRL` *en Windows*
`raptorxmlxbrl` *en Unix (Linux, Mac)*

* Nota: las minúsculas (`raptorxmlxbrl`) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que la mezcla de mayúsculas y minúsculas (`RaptorXMLXBRL`) solo funciona en Windows y Mac.

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+XBRL Server. Esta opción debe escribirse antes del comando: **`raptorxmlxbrl`** `--version licenseserver`.

© 2014 Altova GmbH

3.7 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\RaptorXMLXBRLServer2014\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\RaptorXMLXBRLServer2014\bin\`. El DLL tendrá un nombre similar a este `RaptorXMLXBRLServer_ci.dll`. La parte `_ci` del nombre contiene el código del idioma. Por ejemplo, en `RaptorXMLXBRLServer_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+XBRL 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`.

3.7.1 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).

Windows **RaptorXMLXBRL exportresourcestrings *CódigoIdioma* *ArchivoXMLSalida***
Linux **raptorxmlxbrl exportresourcestrings *CódigoIdioma* *ArchivoXMLSalida***
Mac **raptorxmlxbrl exportresourcestrings *CódigoIdioma* *ArchivoXMLSalida***

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.

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

Uso de mayúsculas/minúsculas en la línea de comandos

RaptorXMLXBRL *en Windows*
raptorxmlxbrl *en Unix (Linux, Mac)*

* Nota: las minúsculas (raptorxmlxbrl) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que la mezcla de mayúsculas y minúsculas (RaptorXMLXBRL) solo funciona en Windows y Mac.

 © 2014 Altova GmbH

3.7.2 setdeflang

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

Windows	**RaptorXMLXBRL setdeflang | sdl *CódigoIdioma***
Linux	**raptorxmlxbrl setdeflang | sdl *CódigoIdioma***
Mac	**raptorxmlxbrl setdeflang | sdl *CódigoIdioma***

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

```
raptorxmlxbrl setdeflang es
```

Uso de mayúsculas/minúsculas en la línea de comandos

RaptorXMLXBRL *en Windows*
raptorxmlxbrl *en Unix (Linux, Mac)*

* Nota: las minúsculas (`raptorxmlxbrl`) funcionan en todas las plataformas (Windows, Linux y Mac), mientras que la mezcla de mayúsculas y minúsculas (`RaptorXMLXBRL`) solo funciona en Windows y Mac.

Idiomas compatibles
Los idiomas disponibles son:

EN	inglés
DE	alemán
ES	español
JA	japonés

3.8 Opciones

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

- Catálogos, recursos globales y archivos ZIP
- Mensajes, errores y ayuda
- Procesamiento
- XBRL
- XML
- XSD
- XQuery
- XSLT

 © 2014 Altova GmbH

3.8.1 Catálogos, recursos globales, archivos ZIP

catalog

--catalog = *ARCHIVO*
Especifica la ruta de acceso absoluta a un archivo de catálogo que no está en el archivo de catálogo raíz instalado. El valor predeterminado es la ruta de acceso absoluta del archivo de catálogo raíz instalado (`<carpeta-instalación>\Altova\RaptorXMLXBRLServer2014\etc\RootCatalog.xml`). Consulte el apartado Catálogos XML para obtener más información.

user-catalog

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

enable-globalresources

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

globalresourceconfig [gc]

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

globalresourcefile [gr]

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

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

3.8.2 Mensajes, errores y ayuda

error-format

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

error-limit

--error-limit = *N*
Especifica el límite de errores. Se permiten los valores comprendidos entre el `1` y el `999`. 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`.

help

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

log-output

--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

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

version

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

 © 2014 Altova GmbH

3.8.3 Procesamiento

listfile

--listfile = true|false

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

parallel-assessment [pa]

--pa | --parallel-assessment = true|false

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

script

--script = *ARCHIVO*

Una vez finalizada la validación, ejecuta el script Python.

streaming

--streaming = true|false

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

3.8.4 XBRL

- Opciones de validación y procesamiento XBRL
  - **dimensions**

    --dimensions = true|false
    Habilita las extensiones XBRL Dimension 1.0. Valor predeterminado: true.
    Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en true.

  - **listfile**

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

  - **parallel-assessment [pa]**

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

  - **preload-xbrl-schemas**

    --preload-xbrl-schemas = true|false
    Carga previa de los esquemas de la especificación XBRL 2.1. Valor predeterminado: true.
    Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en true.

  - **recurse**

    --recurse = true|false
    Esta opción se utiliza para seleccionar ficheros dentro de 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. Valor predeterminado: false

 © 2014 Altova GmbH

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

schema-imports

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

Esta opción indica el comportamiento de los elementos `xs:import`. Cada uno de estos elementos tiene un atributo opcional `namespace` y un atributo opcional `schemaLocation`: `<import namespace="unEspacioNombres" schemaLocation="unaURL">`. 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**.
- `load-by-namespace`: el valor del atributo `namespace` se utiliza para buscar el esquema por medio de una asignación de catálogo.
- `load-combining-both`: si el atributo `namespace` o `schemaLocation` tiene una asignación de catálogo, entonces se usa la asignación. Si ambos atributos tienen asignaciones de catálogo, entonces es el valor de la opción `--schema-mapping` (opción XBRL y opción XML/XSD) decide qué asignación se utiliza. Si no hay ninguna asignación de catálogo, entonces se usa el atributo `schemaLocation`.
- `license-namespace-only`: se importa el espacio de nombres. No se importa el documento de esquema.

schema-location-hints

--schema-location-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 de instancia XML o XBRL.
- 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` (opción XBRL y opción XML/XSD) decide qué asignación se utiliza. Si ni el espacio de nombres ni la URL tiene una asignación de catálogo, se usa la URL.
- El valor `ignore` ignora los atributos `xsi:schemaLocation` y `xsi:noNamespaceSchemaLocation`.

schema-mapping

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

Si 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). Valor predeterminado: `prefer-schemalocation`.

script

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

treat-inconsistencies-as-errors

--treat-inconsistencies-as-errors = `true|false`
La validación XBRL falla si el archivo contiene incoherencias según lo definido por la especificación XBRL 2.1. El valor predeterminado es `false`.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

validate-dts-only

--validate-dts-only = `true|false`
La DTS se detecta iniciando desde el documento de instancia XBRL. Todos los esquemas de taxonomía y bases de enlace referenciados se detectan y validan. Los demás documentos de instancia XBRL se ignoran. Valor predeterminado: `false`.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

xinclude

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

Opciones de fórmulas y aserciones XBRL

assertions-output

--assertions-output = *ARCHIVO*
Escribe el resultado de la evaluación de la aserción en el *ARCHIVO* indicado. Si se define, especifica automáticamente `--formula-execution=true`.

assertions-output-format

--assertions-output-format = `json|xml`
Especifica el formato de salida de la evaluación de la aserción. Valor predeterminado: `json`.

formula

--formula = `true|false`
Habilita las extensiones XBRL Formula 1.0. Valor predeterminado: `true`.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se

establece en `true`.

formula-assertion-set [[OBSOLETA]]

--formula-assertion-set = *VALOR*

Limita la ejecución de fórmulas al conjunto de aserciones dado. Añada la opción varias veces para especificar más de un grupo de aserciones. Su forma abreviada es `--as`. El *VALOR* es el valor del atributo `@id` o un URI con un fragmento XPointer que identifica el recurso. También puede usar los valores especiales `##none` y `##all`.

formula-execution

--formula-execution = `true|false`

Habilita la evaluación de fórmulas XBRL. Valor predeterminado: `true`. Si es `true`, especifica automáticamente `--formula=true`.

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

formula-output

--formula-output = *ARCHIVO*

Escribe el resultado de la evaluación de la fórmula en el *ARCHIVO* indicado. Si se define, especifica automáticamente `--formula-execution=true`.

formula-parameters [[OBSOLETA]]

--formula-parameters = *MATRIZ-JSON*

Especifica los parámetros para la evaluación de fórmulas XBRL en formato JSON en la interfaz de la línea de comandos directamente. Esta opción es **obsoleta** (utilice la opción `FormulaParametersFile`; véase Parámetros de fórmula).

El formato de la matriz JSON sería este:

```
[{"name":"p1","namespace":"http://xbrl.org/2008/
conformance","type":"xs:string","value":"hello world from json"}, … ]
```

Sin embargo, tenga cuidado cuando use caracteres de escape en la línea de comandos. En el shell cmd.exe la opción `formula-parameters` tendría esta sintaxis:

```
--formula-parameters="[{\"name\":\"date\",\"namespace\":\"http://
example.com\",\"type\":\"xs:string\",\"value\":\"2007-12-31\"}]"
```

Puede que en otros sistemas operativos y shells sea necesario utilizar otros caracteres de escape.

formula-parameters-file

--formula-parameters-file = *ARCHIVO*

Especifica un *ARCHIVO* que contiene los parámetros para la evaluación de fórmulas XBRL. Puede tratarse de un archivo XML o de un archivo JSON. Consulte el apartado Parámetros de fórmula para obtener más información

- Formatos de parámetro antiguos (obsoletos)

  Formato de archivo XML:

```
<?xml version="1.0" encoding="utf-8"?>
  <parameters>
    <parameter name="p1" namespace="http://xbrl.org/2008/conformance"
type="xs:string" value="hello world from xml"/>
    ...
```

```
</parameters>
```

Formato JSON:

```
[{"name":"p1","namespace":"http://xbrl.org/2008/
conformance","type":"xs:string","value":"hello world from json"}, … ]
```

La clave `namespace` es opcional.

preload-formula-schemas

`--preload-formula-schemas =` `true|false`

Carga previa de los esquemas de la especificación XBRL Formula 1.0. Valor predeterminado: `false`.

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

process-assertion [a]

`--a | --process-assertion =` *VALOR*

Limita la ejecución de fórmulas a la aserción dada. Añada la opción varias veces para especificar más de una aserción. Su forma abreviada es `--a`. El *VALOR* es el valor del atributo `@id` o un URI con un fragmento XPointer que identifica el recurso. También se pueden usar los valores especiales `##none` y `##all`.

process-assertion-set [as]

`--as | --process-assertion-set =` *VALOR*

Limita la ejecución de fórmulas al conjunto de aserciones dado. Añada la opción varias veces para especificar más de un conjunto de aserciones. Su forma abreviada es `--as`. El ***VALOR*** es el valor del atributo `@id` o un URI con un fragmento XPointer que identifica el recurso. También puede usar los valores especiales `##none` y `##all`.

process-formula [f]

`--f | --process-formula =` *VALOR*

Limita la ejecución de fórmulas a la fórmula dada. Añada la opción varias veces para especificar más de una fórmula. Su forma abreviada es `--as`. El ***VALOR*** es el valor del atributo `@id` o un URI con un fragmento XPointer que identifica el recurso. También puede usar los valores especiales `##none` y `##all`.

variableset-execution-timeout

`--variableset-execution-timeout =` *VALOR*

Se aplica cuando se ejecutan fórmulas (**`--formula-execution=true`**). Especifica el tiempo de espera máximo permitido para ejecutar un conjunto de variables único (una fórmula o un valor o una aserción de existencia o de coherencia). El tiempo de espera se indica en minutos y debe ser un número positivo. El valor predeterminado es 30 minutos. Si un conjunto de variables determinado no termina de ejecutarse antes de cumplirse el tiempo de espera, la ejecución se anula. Aparece un mensaje de error (que se introduce en el registro detallado). No obstante, debe tener en cuenta que la comprobación de tiempo de espera solamente se realiza después de evaluarse cada conjunto de variables (y no durante la ejecución de las expresiones XPath). Por tanto, si una expresión XPath tarda mucho en ejecutarse, puede que se cumpla el tiempo de espera. La ejecución del conjunto de variables se anula solamente cuando se ejecute la evaluación completa del conjunto de variables.

 © 2014 Altova GmbH

Opciones de tabla XBRL

Propuesta de recomendación (no es recomendación final)
La implementación actual de RaptorXML+XBRL Server (publicada en marzo de 2014) sigue la propuesta de recomendación Table Linkbase del 18 de diciembre de 2013 y utiliza el espacio de nombres `http://xbrl.org/PR/2013-12-18/table`. Si antes de que Altova publique una versión nueva de RaptorXML+XBRL Server aparece una recomendación nueva o una especificación final, tenga en cuenta que el espacio de nombres que utiliza RaptorXML+XBRL Server será obsoleto y no cumplirá con la nueva recomendación/especificación. La implementación se actualizará en la versión de RaptorXML+XBRL Server inmediatamente posterior a la publicación de la nueva recomendación/especificación.

preload-table-schemas

--preload-table-schemas = `true|false`
Carga previa de los esquemas de la especificación XBRL Table 1.0. Valor predeterminado: `false`.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

process-table [t]

--t | --process-table = *VALOR*
Limita la ejecución de fórmulas a la taba dada. Añada la opción varias veces para especificar más de una tabla. Su forma abreviada es `--t`. El ***VALOR*** es el valor del atributo `@id` o un URI con un fragmento XPointer que identifica el recurso. También puede usar los valores especiales `##none` y `##all`.

table

--table = `true|false`
Habilita la extensión XBRL Table 1.0. Si es `true`, especifica automáticamente `--formula=true` y `--dimensions=true`. Valor predeterminado: `true`.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

table-elimination

--table-elimination = `true|false`
Habilita la eliminación de columnas/filas de tabla vacías en el formato de salida HTML. Valor predeterminado: `true`.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

table-execution

--table-execution = `true|false`
Habilita la evaluación de tablas XBRL. Valor predeterminado: `false`. Si se especifica la opción `--table-output`, a esta opción se asigna `true`. Si tiene el valor `true`, se especifica automáticamente `--table=true`.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

table-output

--table-output = *ARCHIVO*
Escribe la tabla de salida en el ***ARCHIVO*** indicado. Si se define, especifica automáticamente `--table-execution=true`.

table-output-format

--table-output-format = `xml|html`
Especifica el formato de la tabla de salida. Valor predeterminado: `xml`.

Opciones de etiqueta XBRL

concept-label-linkrole

--concept-label-linkrole = *VALOR*
Especifica la función de vínculo preferida a la hora de representar etiquetas de concepto.

concept-label-role

--cconcept-label-role = *VALOR*
Especifica la función de etiqueta preferida a la hora de representar etiquetas de concepto. La función predeterminada es `http://www.xbrl.org/2003/role/label`.

generic-label-linkrole

--generic-label-linkrole = *VALOR*
Especifica la función de vínculo extendido preferida a la hora de representar etiquetas genéricas.

generic-label-role

--generic-label-role = *VALOR*
Especifica la función de etiqueta preferida a la hora de representar etiquetas genéricas. La función predeterminada es `http://www.xbrl.org/2003/role/label`

label-lang

--label-lang = *VALOR*
Especifica el idioma de etiqueta preferido a la hora de representar etiquetas. Valor predeterminado: `en`

 © 2014 Altova GmbH

3.8.5 XML

assessment-mode

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

dtd

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

namespaces

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

xinclude

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

xml-mode

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

xsd

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

3.8.6 XSD

assessment-mode

--assessment-mode = `lax|strict`

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

namespaces

--namespaces = `true|false`

Habilita el procesamiento preparado para espacios de nombres. Esta opción es muy útil si quiere buscar en la instancia XML errores resultantes de espacios de nombres erróneos. Valor predeterminado: `false`.

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

schema-imports

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

Esta opción indica el comportamiento de los elementos `xs:import`. Cada uno de estos elementos tiene un atributo opcional `namespace` y un atributo opcional `schemaLocation`: `<import namespace="unEspacioNombres" schemaLocation="unaURL">`. 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**.
- `load-by-namespace`: el valor del atributo `namespace` se utiliza para buscar el esquema por medio de una asignación de catálogo.
- `load-combining-both`: si el atributo `namespace` o `schemaLocation` tiene una asignación de catálogo, entonces se usa la asignación. Si ambos atributos tienen asignaciones de catálogo, entonces es el valor de la opción `--schema-mapping` (opción XBRL y opción XML/XSD) decide qué asignación se utiliza. Si no hay ninguna asignación de catálogo, entonces se usa el atributo `schemaLocation`.
- `license-namespace-only`: se importa el espacio de nombres. No se importa el documento de esquema.

schema-location-hints

--schema-location-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 de instancia XML o XBRL.
- 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

 © 2014 Altova GmbH

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` (opción XBRL y opción XML/XSD) decide qué asignación se utiliza. Si ni el espacio de nombres ni la URL tiene una asignación de catálogo, se usa la URL.
- El valor `ignore` ignora los atributos `xsi:schemaLocation` y `xsi:noNamespaceSchemaLocation`.

schema-mapping

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

Si 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). Valor predeterminado: `prefer-schemalocation`.

xsd-version

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

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

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

3.8.7 XQuery

indent-characters

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

input

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

omit-xml-declaration

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

output

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

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

output-indent

--output-indent = `true|false`
Si el valor es `true`, la sangría del documento de salida seguirá su estructura jerárquica. Si el valor es `false`, el documento de salida no tendrá sangría jerárquica. Valor predeterminado: `false`.
Nota: si no se especifica un valor para la opción, el valor booleano de la opción se establece en `true`.

output-method

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

param [p]

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

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

 © 2014 Altova GmbH

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

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

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

XSLT

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

```
raptorxmlxbrl xslt --input=c:\Test.xml --output=c:\Salida.xml --
param=date://node[1]/@att1 --p=title:cadenasinespacios --
param=title:"'cadena con espacios'" --p=amount:456 c:\Test.xslt
```

3.8.8 XSLT

chartext-disable

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

dotnetext-disable

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

indent-characters

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

input

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

javaext-barcode-location

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

javaext-disable

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

output

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.

param [p]

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

XQuery

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

Al usar la palabra clave `external`, `$foo` se convierte en parámetro externo y su valor se pasa en tiempo de ejecución desde una fuente externa. El parámetro externo recibe un

 © 2014 Altova GmbH

valor con el comando de la ILC. Por ejemplo: `--param=foo:'MiNombre'`

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

XSLT

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

```
raptorxmlxbrl xslt --input=c:\Test.xml --output=c:\Salida.xml --
param=date://node[1]/@att1 --p=title:cadenasinespacios --
param=title:"'cadena con espacios'" --p=amount:456 c:\Test.xslt
```

streaming

--streaming = `true|false`

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

template-entry-point

--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

--template-mode = *VALOR*

Especifica el modo de plantilla que debe usarse para la transformación.

xslt-version

--xslt-version = `1|2|3`

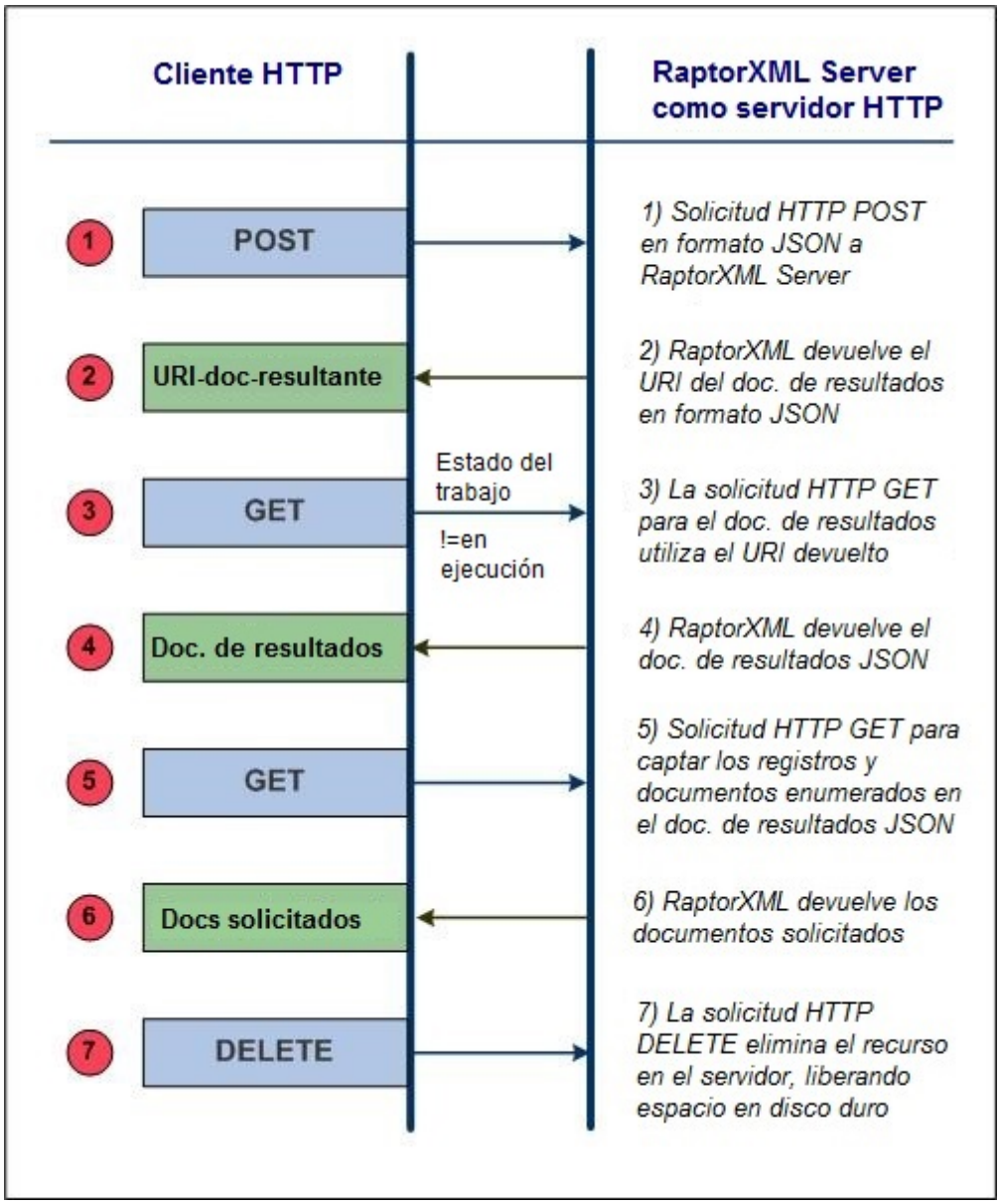
Especifica si el procesador XSLT debe usar XSLT 1.0, XSLT 2.0 o XSLT 3.0. Valor predeterminado: `3`

Altova RaptorXML+XBRL Server 2014

Interfaz HTTP

4 Interfaz HTTP

RaptorXML+XBRL 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.



Problemas de seguridad relacionados con la interfaz HTTP

 © 2014 Altova GmbH

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 instale y configure RaptorXML+XBRL 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+XBRL 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 +XBRL Server. Recuerde que antes debe instalarlo y asignarle licencias.

1. Para poder acceder a RaptorXML+XBRL 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+XBRL 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.

 © 2014 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+XBRL Server se instala por defecto en esta carpeta:

```
<CarpetaArchivosProgramas>\Altova\RaptorXMLXBRLServer2014\bin
\RaptorXMLXBRL.exe
```

Puede usar este ejecutable para iniciar RaptorXML+XBRL Server como servicio.

Iniciar RaptorXML como servicio en Windows
Durante el proceso de instalación se registra RaptorXML+XBRL 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+XBRL Server"`**
- Con el ejecutable de RaptorXML+XBRL Server en una ventana del símbolo del sistema: **`RaptorXMLXBRLServer.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+XBRL Server como servicio:

```
[Debian]   sudo /etc/init.d/raptorxmlxbrlserver start
[Ubuntu]   sudo initctl start raptorxmlxbrlserver
[CentOS]   sudo initctl start raptorxmlxbrlserver
```

```
[RedHat]   sudo initctl start raptorxmlxbrlserver
```

Si necesita detener RaptorXML+XBRL Server, utilice este otro comando:

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

Iniciar RaptorXML como servicio en Mac OS X

Utilice este comando para iniciar RaptorXML+XBRL Server como servicio:

```
sudo launchctl load /Library/LaunchDaemons/
com.altova.RaptorXMLXBRLServer2014.plist
```

Si necesita detener RaptorXML+XBRL Server, utilice este otro comando:

```
sudo launchctl unload /Library/LaunchDaemons/
com.altova.RaptorXMLXBRLServer2014.plist
```

 © 2014 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+XBRL 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+XBRL 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+XBRL 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+XBRL 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\RaptorXMLXBRLServer2014\etc\server_config.xml
```

La configuración inicial para RaptorXML+XBRL 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\RaptorXMLXBRLServer2014\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+XBRL Server como servicio.

Si quiere recrear el archivo de configuración original (para volver a la configuración inicial), ejecute el comando `createconfig`:

```
RaptorXMLXBRL.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

 © 2014 Altova GmbH

A continuación puede ver el archivo de configuración del servidor `server_config.xml` en su versión original. Las opciones de configuración se explican más abajo.

server_config.xml

```
<config xmlns="http://www.altova.com/schemas/altova/raptorxml/config"
  xsi:schemaLocation="http://www.altova.com/schemas/altova/raptorxml/config
  http://www.altova.com/schemas/altova/raptorxml/config.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <language>en</language>
  <server.unrestricted-filesystem-access>true</server.unrestricted-filesystem-
  access>
  <server.output-root-dir>C:\ProgramData\Altova\RaptorXMLXBRLServer2014\Output
  \</server.output-root-dir>
  <server.script-root-dir>C:\Program Files (x86)\Altova\RaptorXMLXBRLServer2014
  \etc\scripts\</server.script-root-dir>
  <!--<server.catalog-file>catalog.xml</server.catalog-file>-->
  <server.log-file>C:\ProgramData\Altova\RaptorXMLXBRLServer2014\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\RaptorXMLXBRLServer2014\Log
  \access.log</http.access-file>
  <http.error-file>C:\ProgramData\Altova\RaptorXMLXBRLServer2014\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\RaptorXMLXBRLServer2014\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+XBRL Server. Si quiere que RaptorXML+XBRL Server acepte conexiones desde equipos remotos, quite la marca de comentario de este elemento y defina su contenido como `0.0.0.0`, es decir: `<http.socket-host>0.0.0.0</http.socket-host>`. Esto almacena el servicio en cada interfaz direccionable del equipo servidor. En este caso, compruebe que las opciones del servidor de seguridad están configuradas correctamente. Las excepciones de entrada del servidor de seguridad para los productos de Altova deben registrarse de la siguiente manera:
- Altova LicenseServer: puerto `8088`
- Altova RaptorXML+XBRL Server: puerto `8087`
- Altova FlowForce Server: puerto `8082`

http.socket-port
El puerto por el que se accede al servicio. El puerto debe ser fijo y conocido para que las solicitudes HTTP se puedan direccionar correctamente al servicio.

http.log-screen

 © 2014 Altova GmbH

Si inició RaptorXML+XBRL Server con el comando `RaptorXMLXBRLServer.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.

La dirección de RaptorXML+XBRL 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+XBRL Server con:

```
http://100.60.300.6:8087/
```

Nota: tras modificar el archivo de configuración del servidor `server_config.xml`, es necesario reiniciar RaptorXML+XBRL Server para que los cambios se apliquen.

Nota: si tiene problemas para conectarse con RaptorXML+XBRL Server, puede que los archivos `http.access-file` y `http.error-file` le ayuden a resolver el problema.

Nota: los mensajes enviados a RaptorXML+XBRL 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

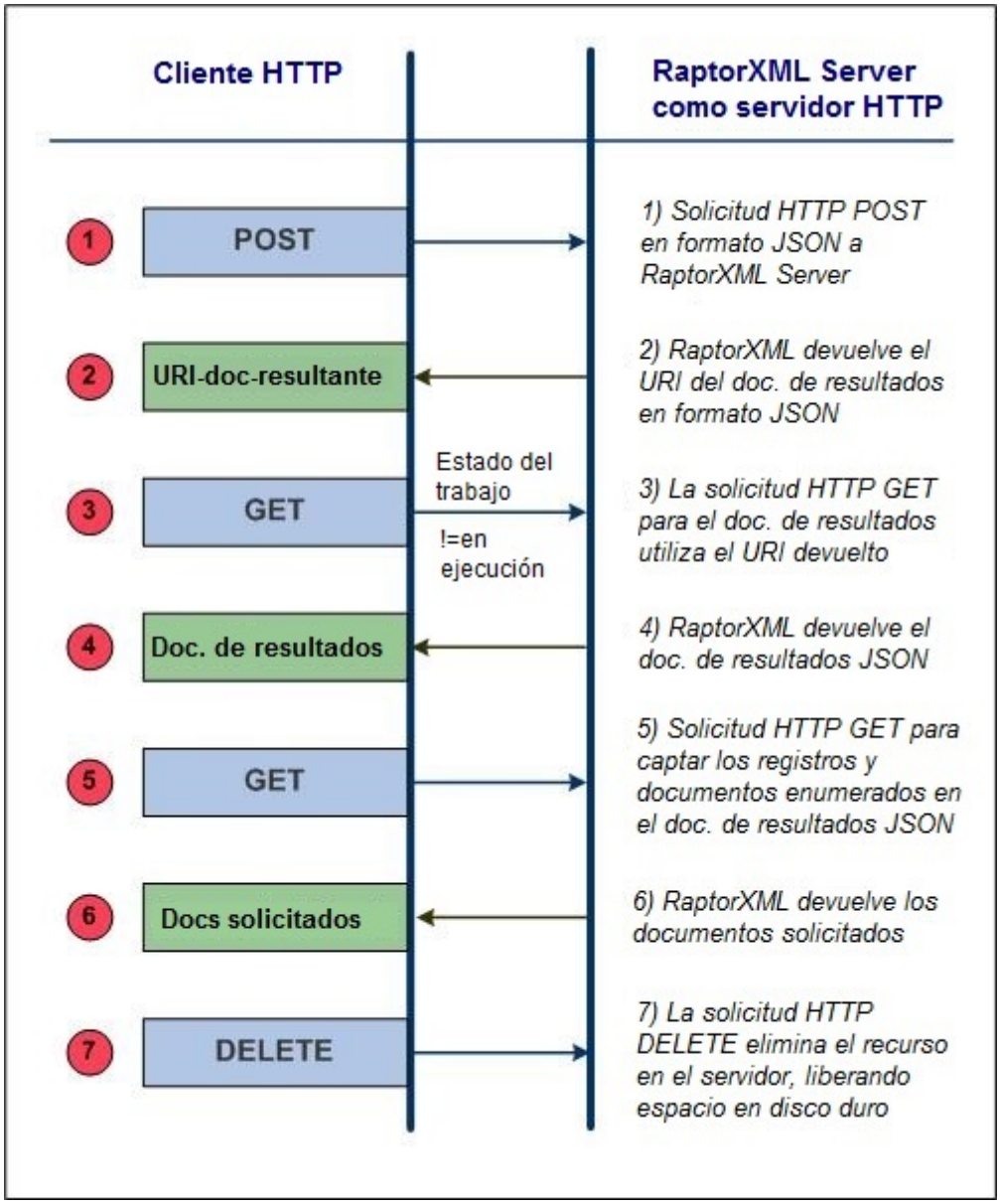
Tras iniciar RaptorXML+XBRL Server como servicio, podrá acceder a sus funciones cualquier cliente HTTP que pueda:

- 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.

 © 2014 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 +XBRL 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+XBRL Server (la dirección inicial del servidor). Dicha solicitud recibe el nombre de **trabajo de RaptorXML+XBRL Server**.

2. Si RaptorXML+XBRL 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+XBRL 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+XBRL 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+XBRL 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.

 © 2014 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+XBRL 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.

A continuación puede ver un ejemplo con opciones. Observe que algunas opciones (como `input` o `xslt-version`) toman un valor directo, mientras que otros (como `param`) toman un par clave-valor y, por tanto, necesitan otra sintaxis.

```
{
   "command": "xslt",
   "args": [
           "file:///C:/Work/Prueba.xslt"
           ],
   "options": {
               "input": "file:///C:/Work/Prueba.xml",
               "xslt-version": 1,
               "param": {
```

 © 2014 Altova GmbH

```
                        "key": "miPruebaParam",
                        "value": "ValorDeParám"
                        },
                "output": "file:///C:/temp/salida2.xml"
                }
}
```

El ejemplo que aparece a continuación muestra otro tipo de opción más: una matriz de valores (como en la opción `xsd` del ejemplo). En casos así, debe usar la sintaxis de una matrix JSON.

```
{
   "command": "xsi",
   "args": [
           "file:///C:/Work/Prueba.xml"
           ],
   "options": {
               "xsd" : ["file:///C:/Work/File1.xsd", "file:///C:/Work/
Archivo2.xsd"]
               }
}
```

Cargar archivos con la solicitud POST

Puede cargar los archivos que deben procesarse dentro del cuerpo de la solicitud `POST`. En este caso, la solicitud `POST` debe crearse como se muestra a continuación.

Encabezado de la solicitud

En el encabezado de la solicitud, defina `Content-Type: multipart/form-data` y especifique una cadena de texto cualquiera como frontera. Por ejemplo:

```
Content-Type: multipart/form-data; boundary=---MiFrontera
```

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

Cuerpo de la solicitud: parte del mensaje

El cuerpo de la solicitud está formado por varias partes `form-data`, separadas por la cadena de frontera especificada en el encabezado (*ver más arriba*):

- *Partes `form-data` obligatorias:* la parte **`msg`**, que especifica la acción de procesamiento solicitada, y la parte **`args`**, que contiene los archivos que deben cargarse como argumentos del comando especificado en la parte **`msg`**. (*Ver ejemplo que aparece más abajo*).
- *Parte `form-data` opcional:* la parte **`additional_files`** contiene los archivos a los que se hace referencia desde las partes `form-data` obligatorias **`msg`** y **`args`**. Además, las

partes `form-data` con nombre de opción de comando también pueden incluir archivos para cargar.

Nota: todos los archivos cargados se crean en el mismo directorio virtual.

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:

```
raptorxmlxbrl 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: attachment; filename="Primero.xml"; name="args"
Content-Type: application/octet-stream

  <?xml version="1.0" encoding="UTF-8"?>                              4
  <test xsi:noNamespaceSchemaLocation="Demo.xsd" xmlns:xsi="http://
  www.w3.org/2001/XMLSchema-instance">42</test>

---FronteraDeParte                                                    5
Content-Disposition: attachment; filename="Segundo.xml"; name="args"
Content-Type: application/octet-stream

  <?xml version="1.0" encoding="UTF-8"?>                              6
  <test xsi:noNamespaceSchemaLocation="Demo.xsd" xmlns:xsi="http://
  www.w3.org/2001/XMLSchema-instance">35</test>

---FronteraDeParte                                                    7
Content-Disposition: attachment; filename="Demo.xsd";
name="additional_files"
Content-Type: application/octet-stream

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

---FronteraDeParte--                                                  9
```

 © 2014 Altova GmbH

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

4 El texto del archivo `Primero.xml`. Contiene una referencia a un esquema llamado `Demo.xsd`, que también se cargará (en la parte `form-data` **`additional_files`**).

5 El segundo miembro de la matriz **`args`** es un archivo adjunto llamado `Segundo.xml`.

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

7 El final del último miembro de la matriz **`args`** (y por tanto de la matriz propiamente dicha) se indica añadiendo el sufijo `'--'` a la cadena de frontera de **`args`**.

8 El texto del archivo `Demo.xsd`.

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

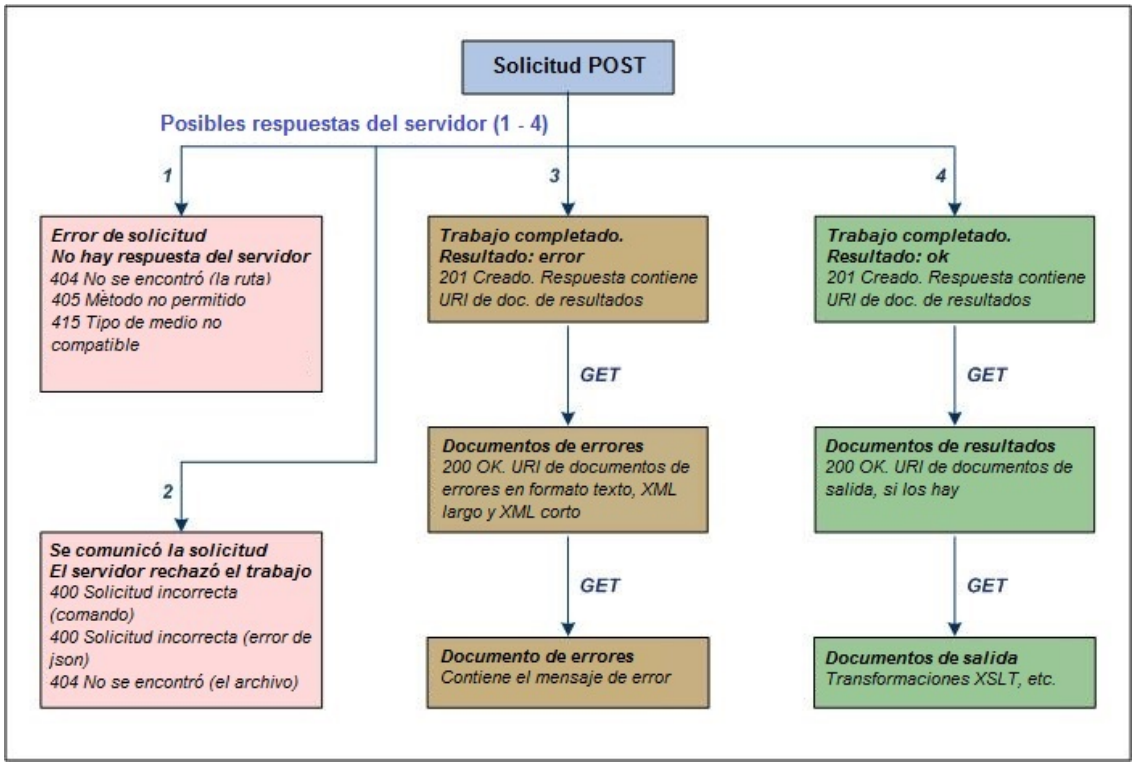
4.2.2 Respuesta del servidor a solicitudes POST

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). 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.

 © 2014 Altova GmbH

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 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.

 © 2014 Altova GmbH

- 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.

 © 2014 Altova GmbH

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+XBRL Server devuelve el documento solicitado.

 © 2014 Altova GmbH

4.2.5 Liberar espacio tras el procesamiento

RaptorXML+XBRL 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.

Altova RaptorXML+XBRL Server 2014

Interfaz Python

5 Interfaz Python

La interfaz Python de RaptorXML+XBRL Server permite acceder y recuperar datos de documentos XML, XSD, instancias XBRL y taxonomías XBRL a través de diferentes API de Python (para XML, XSD y XBRL). 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+XBRL Server.

Las API de Python
Las API de Python (para XML, XSD y XBRL) ofrecen acceso a metadatos, información estructural y datos incluidos en documentos XML, XSD, instancias XBRL y taxonomías XBRL. 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+XBRL Server que escriba datos desde un documento XML o de instancia XBRL 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
- API de Python para XBRL

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)`
- `valxbrltaxonomy (dts)`
- `valxbrl (xbrl)`

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+XBRL 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+XBRL 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.

 © 2014 Altova GmbH

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.

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** como mínimo.

Trabajar con scripts Python seguros
Cuando se especifica un script Python por HTTP para RaptorXML+XBRL 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+XBRL Server
El script Python se pasa a RaptorXML con el parámetro **`--script`** de estos comandos:

- `valxml-withxsd (xsi)`
- `valxsd (xsd)`
- `valxbrltaxonomy (dts)`
- `valxbrl (xbrl)`

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.

 © 2014 Altova GmbH

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+XBRL 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+XBRL Server
valxml-withxsd (xsi)	**on_xsi_valid**(job,instance)
valxsd (xsd)	**on_xsd_valid**(job,schema)
valxbrltaxonomy (dts)	**on_dts_valid**(job,dts)
valxbrl (xbrl)	**on_xbrl_valid**(job,instance)

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, xbrl
```

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`**, **`xbrl`** 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 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+XBRL 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+XBRL 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.

 © 2014 Altova GmbH

5.2 Ejecutar scripts Python

Para pasar scripts Python a RaptorXML+XBRL 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)`
- `valxbrltaxonomy (dts)`
- `valxbrl (xbrl)`

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:

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

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+XBRL 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.

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:

```
raptorxmlxbrl 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

 © 2014 Altova GmbH

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:

```
raptorxmlxbrl 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>
```

 © 2014 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:

```
raptorxmlxbrl 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:

```
raptorxmlxbrl 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:
```

 © 2014 Altova GmbH

```
        prefix = ''

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


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)
```

 © 2014 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>
```

 © 2014 Altova GmbH

```
                    <Shares>
                           1500
                    </Shares>
                    <LeaveTotal>
                           25
                    </LeaveTotal>
                    <LeaveUsed>
                           4
                    </LeaveUsed>
                    <LeaveLeft>
                           21
                    </LeaveLeft>
                    <union>
                           3
                    </union>
                    <list>
                           abc def
                    </list>
                    <bool>
                           true
                    </bool>
                    <idref>
                           fred
                    </idref>
                    <idrefs>
                           fred joe
                    </idrefs>
                    <entity>
                           myUnparsedEntity
                    </entity>
                    <notation>
                           Altova-Orgchart
                    </notation>
                </Person>
                ...
            </Department>
            ...
        </Office>
        ...
</OrgChart>
```

5.5 Ejemplo de script 3: informe XBRL

Este script Python procesa los datos de cualquier taxonomía XBRL y crea un documento de salida llamado **`report.html`** que incluye una lista con todos los elementos de tipo concepto y todas las tuplas de la taxonomía.

El script se pasa en la ILC con este comando:

```
raptorxmlxbrl dts --script=dtsreport.py AnyTaxonomy.xsd
```

En este apartado puede encontrar:

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

 © 2014 Altova GmbH

5.5.1 Script

El script Python con anotaciones que aparece a continuación procesa los datos de cualquier taxonomía XBRL y crea una lista con todos los elementos de tipo concepto y todas las tuplas de la taxonomía. El documento de salida se llama **report.html**.

El script se pasa en la ILC con este comando:

```
raptorxmlxbrl dts --script=dtsreport.py AnyTaxonomy.xsd
```

Nombre del archivo: dtsreport.py

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

def getBalance(item):
    """Devolver el saldo en forma de cadena para el elemento de tipo concepto
dado"""

    if item.balance == xbrl.Concept.DEBIT:
        return 'Debit'
    elif item.balance == xbrl.Concept.CREDIT:
        return 'Credit'
    else:
        return 'None'


def getPeriodType(item):
    """Devolver el tipo de período en forma de cadena para el elemento de tipo
concepto dado"""

    if item.period_type == xbrl.Concept.INSTANT:
        return 'Instant'
    elif item.period_type == xbrl.Concept.DURATION:
        return 'Duration'
    else:
        return 'None'


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

    text = ''
    # Iterar en todos los nodos secundarios y concatenar todos los nodos de
caracteres
    for child in elem.children:
        if isinstance(child,xml.Character):
            text += child.character_code
    return text


def getLabel(concept):
    """Devolver el texto de la primera etiqueta conectada a este concepto"""
```

```
    for label in concept.label_elements:
        # Devolver el valor de texto del primer elemento de etiqueta conectado
        return getElemTextValue(label)
    # Si no hay etiquetas conectadas a este concepto, devolver un espacio de no
    separación
    return ' '


def writeItem(f,item):
    """Escribir algo de información sobre el elemento de tipo concepto"""

    f.write('<h3>'+item.qname.local_name+'</h3>\n')
    f.write('<p><table border="1">\n')
    f.write('<tr><td>Name</td><td>'+item.qname.local_name+'</td></tr>')
    f.write('<tr><td>Namespace</td><td>'+item.qname.namespace_name+'</td></tr>')
    f.write('<tr><td>Type</td><td>'+item.element_declaration.type_definition.name
    +'</td></tr>')
    f.write('<tr><td>Abstract</td><td>'+str(item.is_abstract())+'</td></tr>')
    f.write('<tr><td>Nillable</td><td>'+str(item.is_nillable())+'</td></tr>')
    f.write('<tr><td>Numeric</td><td>'+str(item.is_numeric())+'</td></tr>')
    f.write('<tr><td>Balance</td><td>'+getBalance(item)+'</td></tr>')
    f.write('<tr><td>Period Type</td><td>'+getPeriodType(item)+'</td></tr>')
    f.write('<tr><td>Label</td><td>'+getLabel(item)+'</td></tr>')
    f.write('</table></p>\n')


def writeTuple(f,tuple):
    """Escribir algo de información sobre la tupla"""

    f.write('<h3>'+tuple.qname.local_name+'</h3>\n')
    f.write('<p><table border="1">\n')
    f.write('<tr><td>Name</td><td>'+tuple.qname.local_name+'</td></tr>')
    f.write('<tr><td>Namespace</td><td>'+tuple.qname.namespace_name+'</td></tr>')
    f.write('<tr><td>Abstract</td><td>'+str(tuple.is_abstract())+'</td></tr>')
    f.write('<tr><td>Nillable</td><td>'+str(tuple.is_nillable())+'</td></tr>')
    f.write('<tr><td>Label</td><td>'+getLabel(tuple)+'</td></tr>')
    f.write('</table></p>\n')


def writeReport(dts,filename):
    """Escribir un informe con todos los elementos de tipo concepto y tupla de la
    taxonomía"""

    # Abrir archivo de salida
    f = open(filename,'w')
    f.write('<html><title>Report</title><body>\n')
    # Escribir todos los elementos de tipo concepto
    f.write('<h1><center>Item Concepts</center></h1>\n')
    for item in dts.items:
        writeItem(f,item)
    # Escribir todos los conceptos de tipo tupla
    f.write('<h1><center>Tuple Concepts</center></h1>\n')
    for tuple in dts.tuples:
        writeTuple(f,tuple)
    f.write('</body></html>\n')
```

 © 2014 Altova GmbH

```
    # Cerrar el archivo de salida
    f.close()


def on_dts_valid(job,dts):
    """Este método será invocado automáticamente por RaptorXMLXBRL una vez
    finalizada la validación de la taxonomía XBRL"""

    # Crear el archivo 'report.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,'report.html')
    # Crear informe en formato HTML para la DTS
    writeReport(dts,filename)
    # Registrar el archivo de salida 'report.html' recién generado
    job.append_output_filename(filename)
```

5.5.2 Documento de resultados

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

Nombre del archivo: report.html

```
<html><title>Report</title><body>

<h1><center>Item Concepts</center></h1>

<h3>street</h3>
<p><table border="1">
<tr><td>Name</td><td>street</td></tr><tr><td>Namespace</td><td>http://
www.example.com/test</td></tr><tr><td>Type</td><td>stringItemType</td></
tr><tr><td>Abstract</td><td>False</td></tr><tr><td>Nillable</td><td>False</td></
tr><tr><td>Numeric</td><td>False</td></tr><tr><td>Balance</td><td>None</td></
tr><tr><td>Period Type</td><td>Instant</td></tr><tr><td>Label</td><td> </
td></tr></table></p>

<h3>city</h3>
<p><table border="1">
<tr><td>Name</td><td>city</td></tr><tr><td>Namespace</td><td>http://
www.example.com/test</td></tr><tr><td>Type</td><td>stringItemType</td></
tr><tr><td>Abstract</td><td>False</td></tr><tr><td>Nillable</td><td>False</td></
tr><tr><td>Numeric</td><td>False</td></tr><tr><td>Balance</td><td>None</td></
tr><tr><td>Period Type</td><td>Instant</td></tr><tr><td>Label</td><td> </
td></tr></table></p>

<h3>stateOrProvince</h3>
<p><table border="1">
<tr><td>Name</td><td>stateOrProvince</td></tr><tr><td>Namespace</td><td>http://
www.example.com/test</td></tr><tr><td>Type</td><td>stringItemType</td></
tr><tr><td>Abstract</td><td>False</td></tr><tr><td>Nillable</td><td>False</td></
tr><tr><td>Numeric</td><td>False</td></tr><tr><td>Balance</td><td>None</td></
tr><tr><td>Period Type</td><td>Instant</td></tr><tr><td>Label</td><td> </
td></tr></table></p>

<h3>country</h3>
<p><table border="1">
<tr><td>Name</td><td>country</td></tr><tr><td>Namespace</td><td>http://
www.example.com/test</td></tr><tr><td>Type</td><td>stringItemType</td></
tr><tr><td>Abstract</td><td>False</td></tr><tr><td>Nillable</td><td>False</td></
tr><tr><td>Numeric</td><td>False</td></tr><tr><td>Balance</td><td>None</td></
tr><tr><td>Period Type</td><td>Instant</td></tr><tr><td>Label</td><td> </
td></tr></table></p>

<h3>zipOrPostalCode</h3>
<p><table border="1">
<tr><td>Name</td><td>zipOrPostalCode</td></tr><tr><td>Namespace</td><td>http://
www.example.com/test</td></tr><tr><td>Type</td><td>stringItemType</td></
tr><tr><td>Abstract</td><td>False</td></tr><tr><td>Nillable</td><td>False</td></
tr><tr><td>Numeric</td><td>False</td></tr><tr><td>Balance</td><td>None</td></
tr><tr><td>Period Type</td><td>Instant</td></tr><tr><td>Label</td><td> </
```

```
td></tr></table></p>

<h1><center>Tuple Concepts</center></h1>
<h3>address</h3>
<p><table border="1">
<tr><td>Name</td><td>address</td></tr><tr><td>Namespace</td><td>http://
www.example.com/test</td></tr><tr><td>Abstract</td><td>False</td></
tr><tr><td>Nillable</td><td>False</td></tr><tr><td>Label</td><td> </td></
tr></table></p>

<h3>anotherAddress</h3>
<p><table border="1">
<tr><td>Name</td><td>anotherAddress</td></tr><tr><td>Namespace</td><td>http://
www.example.com/test</td></tr><tr><td>Abstract</td><td>False</td></
tr><tr><td>Nillable</td><td>False</td></tr><tr><td>Label</td><td> </td></
tr></table></p>

</body></html>
```

5.6 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.

 © 2014 Altova GmbH

5.7 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.

 © 2014 Altova GmbH

5.7.1 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`.

5.7.2 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 binario 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`.

 © 2014 Altova GmbH

5.7.3 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`.

5.7.4 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.

 © 2014 Altova GmbH

5.7.5 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`.

 © 2014 Altova GmbH

5.7.6 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.

5.7.7 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.

 © 2014 Altova GmbH

5.7.8 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`.

5.7.9 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`.

 © 2014 Altova GmbH

5.7.10 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.

5.7.11 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`.

 © 2014 Altova GmbH

5.7.12 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`.

5.8 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.

 © 2014 Altova GmbH

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

clase xsd.Defined
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.

 © 2014 Altova GmbH

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 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
```

 © 2014 Altova GmbH

5.8.1 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.

5.8.2 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.

 © 2014 Altova GmbH

5.8.3 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.

5.8.4 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.

 © 2014 Altova GmbH

5.8.5 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.

5.8.6 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.

 © 2014 Altova GmbH

5.8.7 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.

 © 2014 Altova GmbH

5.8.8 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.

5.8.9 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`

 © 2014 Altova GmbH

5.8.10 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`.

5.8.11 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.

 © 2014 Altova GmbH

5.8.12 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__()`

5.8.13 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`

 © 2014 Altova GmbH

5.8.14 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.

5.8.15 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.

 © 2014 Altova GmbH

5.8.16 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*.

 © 2014 Altova GmbH

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*.

5.8.17 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`

 © 2014 Altova GmbH

5.8.18 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.

5.8.19 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.

 © 2014 Altova GmbH

5.8.20 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

5.8.21 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__()

 © 2014 Altova GmbH

5.8.22 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`.

5.8.23 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`

 © 2014 Altova GmbH

5.8.24 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`.

5.8.25 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.

 © 2014 Altova GmbH

5.8.26 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

5.8.27 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

 © 2014 Altova GmbH

5.8.28 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

5.8.29 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`

 © 2014 Altova GmbH

5.8.30 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

5.8.31 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.

 © 2014 Altova GmbH

5.8.32 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.

5.8.33 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

 © 2014 Altova GmbH

5.8.34 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.

5.8.35 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`.

 © 2014 Altova GmbH

5.8.36 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.

5.8.37 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()

 © 2014 Altova GmbH

Ofrece declaraciones de notación.

`Schema.resolve_identity_constraint_definition()`
Ofrece definiciones de restricción de identidad.

5.8.38 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.

 © 2014 Altova GmbH

5.8.39 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__()`

5.8.40 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

 © 2014 Altova GmbH

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.

`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.

5.8.41 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.

 © 2014 Altova GmbH

5.8.42 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.

5.8.43 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__()`

 © 2014 Altova GmbH

5.8.44 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.

5.8.45 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.

 © 2014 Altova GmbH

5.8.46 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`**.

5.8.47 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.

 © 2014 Altova GmbH

5.8.48 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 binario. Para más información sobre tipos de datos, consulte su descripción en la sección *Primitive Datatypes* de la especificación XML Schema.

5.8.49 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.

 © 2014 Altova GmbH

5.8.50 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.

5.8.51 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.

 © 2014 Altova GmbH

5.8.52 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.

5.8.53 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*:
  `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`

5.9 API de Python para XBRL

El módulo **xbrl** ofrece una interfaz Python a la implementación C++ de la capa del modelo de datos XBRL.

Propuesta de recomendación (no es recomendación final)
La implementación actual de RaptorXML+XBRL Server (publicada en marzo de 2014) sigue la propuesta de recomendación Table Linkbase del 18 de diciembre de 2013 y utiliza el espacio de nombres `http://xbrl.org/PR/2013-12-18/table`. Si antes de que Altova publique una versión nueva de RaptorXML+XBRL Server aparece una recomendación nueva o una especificación final, tenga en cuenta que el espacio de nombres que utiliza RaptorXML+XBRL Server será obsoleto y no cumplirá con la nueva recomendación/especificación. La implementación se actualizará en la versión de RaptorXML+XBRL Server inmediatamente posterior a la publicación de la nueva recomendación/especificación.

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

class xbrl.BreakdownResource
La clase `BreakdownResource` representa un recurso de desglose en la base de enlaces de tabla.

class xbrl.Concept
La clase `Concept` representa un concepto XBRL en la DTS.

class xbrl.ConceptAspectValue
La clase `ConceptAspectValue` representa un valor para el aspecto concepto.

class xbrl.ConstraintSet
La clase `ConstraintSet` representa un conjunto de restricciones para los aspectos del modelo de aspecto dimensional.

class xbrl.Context
La clase `Context` representa un contexto XBRL en un documento de instancia.

class xbrl.DefinitionNodeResource
La clase `DefinitionNodeResource` representa un recurso `definitionNode` en la base de enlaces de tabla.

class xbrl.DTS
La clase `DTS` representa una DTS (colección de taxonomías XBRL por descubrir).

class xbrl.Entity
La clase `Entity` representa la parte de entidad de un contexto XBRL.

class xbrl.EntityIdentifier
La clase `EntityIdentifier` representa la parte de identificador de identidad de un contexto XBRL.

 © 2014 Altova GmbH

class xbrl.EntityIdentifierAspectValue
La clase `EntityIdentifierAspectValue` representa un valor para el aspecto identificador de identidad.

class xbrl.Error
La clase `Error` es la clase base para todos los errores relacionados con XBRL.

class xbrl.ExplicitDimensionAspectValue
La clase `ExplicitDimensionAspectValue` representa un valor para el aspecto dimensión.

class xbrl.Fact
La clase `Fact` representa un elemento hecho en un documento de instancia XBRL.

class xbrl.FactSet
La clase `FactSet` representa un conjunto de hechos de instancia XBRL.

class xbrl.Fraction
La clase `Fraction` representa un valor fraccionario de un hecho XBRL de tipo.

class xbrl.Instance
La clase `Instance` representa un documento de instancia XBRL.

class xbrl.LabelResource
La clase `LabelResource` representa un recurso de etiqueta.

class xbrl.LayoutCell
La clase `LayoutCell` representa una celda de la tabla generada.

class xbrl.LayoutDataCell
La clase `LayoutDataCell` representa una celda de datos de la tabla generada.

class xbrl.LayoutHeaderCell
La clase `LayoutHeaderCell` representa una celda de encabezado de la tabla generada.

class xbrl.LayoutRow
La clase `LayoutRow` representa una fila de la tabla generada.

class xbrl.LayoutTable
La clase `LayoutTable` representa la tabla después de aplicar el proceso de resolución y diseño.

class xbrl.LayoutTableSet
La clase `LayoutTableSet` representa el conjunto de tablas después de aplicar el proceso de resolución y diseño.

class xbrl.Period
La clase `Period` representa la parte de período del contexto XBRL.

class xbrl.PeriodAspectValue
La clase `PeriodAspectValue` representa un valor del aspecto período.

class xbrl.ReferenceResource

La clase `ReferenceResource` representa un recurso de referencia.

class xbrl.Resource
La clase `Resource` representa un elemento recurso XLink dentro de un vínculo extendido.

class xbrl.ScenarioAspectValue
La clase `ScenarioAspectValue` representa un valor para el aspecto escenario no XDT.

class xbrl.SegmentAspectValue
La clase `SegmentAspectValue` representa un valor para el aspecto segmento no XDT.

class xbrl.TableError
La clase `TableError` representa un error que se emite durante el proceso de resolución o diseño de tablas.

class xbrl.TableResource
La clase `TableResource` representa un recurso de tabla en una base de enlaces de tabla.

class xbrl.TypedDimensionAspectValue
La clase `TypedDimensionAspectValue` representa un valor para el aspecto dimensión.

class xbrl.Unit
La clase `Unit` representa una unidad XBRL.

class xbrl.UnitAspectValue
La clase `UnitAspectValue` representa un valor para el aspecto unidad.

 © 2014 Altova GmbH

5.9.1 xbrl.BreakdownResource

clase **xbrl.BreakdownResource(xbrl.Resource)**

La clase BreakdownResource representa un recurso de desglose en una base de enlaces de tabla (http://www.xbrl.org/specification/table-linkbase/pr-2013-12-18/table-linkbase-pr-2013-12-18.html#sec-definition-breakdowns).

5.9.2 xbrl.Concept

clase **xbrl.Concept**
La clase Concept representa un concepto XBRL de la DTS.

La clase Concept ofrece estas **constantes**:

Concept.ITEM
Denota que el concepto está en el grupo de sustitución del concepto XBRL `xbrli:item`.

Concept.TUPLE
Denota que el concepto está en el grupo de sustitución del concepto XBRL `xbrli:tuple`.

Concept.HYPERCUBE
Denota que el concepto está en el grupo de sustitución del concepto XBRL `xbrli:hypercube`.

Concept.DIMENSION_EXPLICIT
Denota que el concepto está en el grupo de sustitución del concepto XBRL `xbrli:dimension` y que no tiene definido el atributo `xbrldt:typedDomainRef`

Concept.DIMENSION_TYPED
Denota que el concepto está en el grupo de sustitución del concepto XBRL `xbrli:dimension` y que tiene definido el atributo `xbrldt:typedDomainRef`.

Concept.DEBIT
Denota que el concepto tiene definido el atributo `xbrli:balance` con el valor `debit`.

Concept.CREDIT
Denota que el concepto tiene definido el atributo `xbrli:balance` con el valor `credit`.

Concept.INSTANT
Denota que el concepto tiene definido el atributo `xbrli:periodType` con el valor `instant`.

Concept.DURATION
Denota que el concepto tiene definido el atributo `xbrli:periodType` con el valor `duration`.

Concept.DECIMAL_ITEM_TYPE
Denota que la definición de tipo del concepto es `xbrli:decimalItemType` o una definición de tipo derivada de esta.

Concept.FLOAT_ITEM_TYPE
Denota que la definición de tipo del concepto es `xbrli:floatItemType` o una definición de tipo derivada de esta.

Concept.DOUBLE_ITEM_TYPE
Denota que la definición de tipo del concepto es `xbrli:doubleItemType` o una definición de tipo derivada de esta.

Concept.INTEGER_ITEM_TYPE
Denota que la definición de tipo del concepto es `xbrli:integerItemType` o una definición de

 © 2014 Altova GmbH

tipo derivada de esta.

Concept.NON_POSITIVE_INTEGER_ITEM_TYPE
Denota que la definición de tipo del concepto es `xbrli:nonPositiveIntegerItemType` o una definición de tipo derivada de esta.

Concept.NEGATIVE_INTEGER_ITEM_TYPE
Denota que la definición de tipo del concepto es `xbrli:negativeIntegerItemType` o una definición de tipo derivada de esta.

Concept.LONG_ITEM_TYPE
Denota que la definición de tipo del concepto es `xbrli:longItemType` o una definición de tipo derivada de esta.

Concept.INT_ITEM_TYPE
Denota que la definición de tipo del concepto es `xbrli:intItemType` o una definición de tipo derivada de esta.

Concept.SHORT_ITEM_TYPE
Denota que la definición de tipo del concepto es `xbrli:shortItemType` o una definición de tipo derivada de esta.

Concept.BYTE_ITEM_TYPE
Denota que la definición de tipo del concepto es `xbrli:byteItemType` o una definición de tipo derivada de esta.

Concept.NON_NEGATIVE_INTEGER_ITEM_TYPE
Denota que la definición de tipo del concepto es `xbrli:nonNegativeIntegerItemType` o una definición de tipo derivada de esta.

Concept.UNSIGNED_LONG_ITEM_TYPE
Denota que la definición de tipo del concepto es `xbrli:unsignedLongItemType` o una definición de tipo derivada de esta.

Concept.UNSIGNED_INT_ITEM_TYPE
Denota que la definición de tipo del concepto es `xbrli:unsignedIntItemType` o una definición de tipo derivada de esta.

Concept.UNSIGNED_SHORT_ITEM_TYPE
Denota que la definición de tipo del concepto es `xbrli:unsignedShortItemType` o una definición de tipo derivada de esta.

Concept.UNSIGNED_BYTE_ITEM_TYPE
Denota que la definición de tipo del concepto es `xbrli:unsignedByteItemType` o una definición de tipo derivada de esta.

Concept.POSITIVE_INTEGER_ITEM_TYPE
Denota que la definición de tipo del concepto es `xbrli:positiveIntegerItemType` o una definición de tipo derivada de esta.

Concept.MONETARY_ITEM_TYPE
Denota que la definición de tipo del concepto es `xbrli:monetaryItemType` o una definición de tipo derivada de esta.

Concept.SHARES_ITEM_TYPE
Denota que la definición de tipo del concepto es `xbrli:sharesItemType` o una definición de tipo derivada de esta.

Concept.PURE_ITEM_TYPE
Denota que la definición de tipo del concepto es `xbrli:pureItemType` o una definición de tipo derivada de esta.

Concept.FRACTION_ITEM_TYPE
Denota que la definición de tipo del concepto es `xbrli:fractionItemType` o una definición de tipo derivada de esta.

Concept.STRING_ITEM_TYPE
Denota que la definición de tipo del concepto es `xbrli:stringItemType` o una definición de tipo derivada de esta.

Concept.BOOLEAN_ITEM_TYPE
Denota que la definición de tipo del concepto es `xbrli:booleanItemType` o una definición de tipo derivada de esta.

Concept.HEXBINARY_ITEM_TYPE
Denota que la definición de tipo del concepto es `xbrli:hexBinaryItemType` o una definición de tipo derivada de esta.

Concept.BASE64BINARY_ITEM_TYPE
Denota que la definición de tipo del concepto es `xbrli:base64BinaryItemType` o una definición de tipo derivada de esta.

Concept.ANYURI_ITEM_TYPE
Denota que la definición de tipo del concepto es `xbrli:anyURIItemType` o una definición de tipo derivada de esta.

Concept.QNAME_ITEM_TYPE
Denota que la definición de tipo del concepto es `xbrli:QNameItemType` o una definición de tipo derivada de esta.

Concept.DURATION_ITEM_TYPE
Denota que la definición de tipo del concepto es `xbrli:durationItemType` o una definición de tipo derivada de esta.

Concept.DATETIME_ITEM_TYPE
Denota que la definición de tipo del concepto es `xbrli:dateItemItemType` o una definición de tipo derivada de esta.

Concept.TIME_ITEM_TYPE
Denota que la definición de tipo del concepto es `xbrli:timeItemType` o una definición de tipo derivada de esta.

Concept.DATE_ITEM_TYPE
Denota que la definición de tipo del concepto es `xbrli:dateItemType` o una definición de tipo derivada de esta.

 © 2014 Altova GmbH

Concept.GYEARMONTH_ITEM_TYPE
Denota que la definición de tipo del concepto es `xbrli:gYearMonthItemType` o una definición de tipo derivada de esta.

Concept.GYEAR_ITEM_TYPE
Denota que la definición de tipo del concepto es `xbrli:gYearItemType` o una definición de tipo derivada de esta.

Concept.GMONTHDAY_ITEM_TYPE
Denota que la definición de tipo del concepto es `xbrli:gMonthDayItemType` o una definición de tipo derivada de esta.

Concept.GDAY_ITEM_TYPE
Denota que la definición de tipo del concepto es `xbrli:gDayItemType` o una definición de tipo derivada de esta.

Concept.GMONTH_ITEM_TYPE
Denota que la definición de tipo del concepto es `xbrli:gMonthItemType` o una definición de tipo derivada de esta.

Concept.NORMALIZED_STRING_ITEM_TYPE
Denota que la definición de tipo del concepto es `xbrli:normalizedStringItemType` o una definición de tipo derivada de esta.

Concept.TOKEN_ITEM_TYPE
Denota que la definición de tipo del concepto es `xbrli:tokenItemType` o una definición de tipo derivada de esta.

Concept.LANGUAGE_ITEM_TYPE
Denota que la definición de tipo del concepto es `xbrli:languageItemType` o una definición de tipo derivada de esta.

Concept.NAME_ITEM_TYPE
Denota que la definición de tipo del concepto es `xbrli:NameItemType` o una definición de tipo derivada de esta.

Concept.NCNAME_ITEM_TYPE
Denota que la definición de tipo del concepto es `xbrli:NCNameItemType` o una definición de tipo derivada de esta.

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

Concept.element
Devuelve un objeto `xml.Element` que representa el elemento de información `element` de la declaración de elemento de esquema del concepto.

Concept.element_declaration
Devuelve un objeto `xsd.ElementDeclaration` que representa la declaración de elemento de esquema del concepto.

Concept.qname
Devuelve un objeto xml.QName que representa el nombre completo XML del concepto.

Concept.balance
Devuelve DEBIT o CREDIT si la declaración de elemento de esquema del concepto contiene un atributo xbrli:balance. De lo contrario el atributo es None.

Concept.period_type
Devuelve INSTANT o DURATION si la declaración de elemento de esquema del concepto contiene un atributo xbrli:periodType. De lo contrario el atributo es None.

Concept.concept_type
Devuelve ITEM, TUPLE, HYPERCUBE, DIMENSION_EXPLICIT o DIMENSION_TYPED dependiendo de la afiliación de grupo de sustitución de la declaración de elemento de esquema del concepto.

Concept.item_type
Devuelve el tipo integrado XBRL del cual se deriva la definición de tipo del concepto. Por ejemplo: STRING_ITEM_TYPE, MONETARY_ITEM_TYPE, SHARES_ITEM_TYPE.

Concept.label_elements
Devuelve un objeto generador de objetos xml.Element que representa todos los recursos de etiqueta asociados con este concepto.

Concept.labels
Devuelve un objeto generador de objetos xbrl.LabelResource por cada etiqueta que tenga una relación etiqueta-concepto (http://www.xbrl.org/2003/arcrole/concept-label) con este concepto.

Concept.reference_elements
Devuelve un objeto generador de objetos xml.Element que representa todos los recursos de referencia asociados con este concepto.

Concept.references
Devuelve un objeto generador de objetos xbrl.ReferenceResource por cada etiqueta que tenga una relación referencia-concepto (http://www.xbrl.org/2003/arcrole/concept-reference) con este concepto.

La clase **Concept** ofrece estos **métodos de instancia**:

Concept.is_numeric()
Devuelve True si la definición de tipo del concepto se deriva de uno de los tipos numéricos XBRL integrados.

Concept.is_non_numeric()
Devuelve True si la definición de tipo del concepto se deriva de uno de los tipos no numéricos XBRL integrados.

Concept.is_fraction()
Devuelve True si la definición de tipo del concepto se deriva de xbrli:fractionItemType.

 © 2014 Altova GmbH

`Concept.is_abstract()`

Devuelve `True` si la declaración de elemento de esquema del concepto tiene la propiedad de componente abstracto definida con un valor de `True`.

`Concept.is_nillable()`

Devuelve `True` si la declaración de elemento de esquema del concepto tiene la propiedad de componente "nillable" definida con un valor de `True`.

5.9.3 xbrl.ConceptAspectValue

clase **xbrl.ConceptAspectValue**
La clase ConceptAspectValue represent un valor para el aspecto concepto. Use xbrl.ConstraintSet para crear objetos nuevos de esta clase.

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

ConceptAspectValue.qname
Devuelve un objeto xml.QName que representa el QName del concepto.

ConceptAspectValue.concept
Devuelve un objeto xbrl.Concept.

 © 2014 Altova GmbH

5.9.4 xbrl.ConstraintSet

clase **xbrl.ConstraintSet(Instance instance)**
La clase ConstraintSet representa un conjunto de restricciones para los aspectos del modelo de aspecto dimensional. Por cada aspecto, se puede especificar un aspecto o ninguno.

La clase ConstraintSet ofrece este **constructor**:

ConstraintSet.ConstraintSet
Crea un objeto xbrl.ConstraintSet nuevo vacío que se puede usar para filtrar la instancia xbrl.Instance.

La clase ConstraintSet ofrece estos **atributos de instancia**:

ConstraintSet.concept_aspect
Devuelve un objeto xbrl.ConceptAspectValue que representa la restricción para el aspecto concepto o None si no se especifica ninguna restricción.

ConstraintSet.entity_identifier_aspect
Devuelve un objeto xbrl.EntityIdentifierAspectValue que representa la restricción para el aspecto identificador de identidad o None si no se especifica ninguna restricción.

ConstraintSet.period_aspect
Devuelve un objeto xbrl.PeriodAspectValue que representa la restricción para el aspecto período o None si no se especifica ninguna restricción.

ConstraintSet.scenario_aspect
Devuelve un objeto xbrl.ScenarioAspectValue que representa la restricción para el aspecto escenario no XDT o None si no se especifica ninguna restricción.

ConstraintSet.segment_aspect
Devuelve un objeto xbrl.SegmentAspectValue que representa la restricción para el aspecto segmento no XDT o None si no se especifica ninguna restricción.

ConstraintSet.unit_aspect
Devuelve un objeto xbrl.UnitAspectValue que representa la restricción para el aspecto unidad o None si no se especifica ninguna restricción.

ConstraintSet.dimension_aspects
Devuelve un objeto generador de objetos xbrl.ExplicitDimensionAspectValue y xbrl.TypedDimensionAspectValue que representa las restricciones para el aspecto dimensión.

La clase ConstraintSet ofrece estos **métodos de instancia**:

ConstraintSet.copy()
Devuelve un objeto xbrl.ConstraintSet nuevo que contiene las mismas restricciones.

ConstraintSet.clear()
Quita todas las restricciones de todos los aspectos.

ConstraintSet.append(aspectValue)
Añade una restricción nueva. Las restricciones para este aspecto se sobrescriben.

ConstraintSet.extend(constraintSet)
Añade todas las restricciones de conjunto de restricciones `xbrl.ConstraintSet`. Las restricciones se sobrescriben.

ConstraintSet.set_concept_aspect(concept)
Establece una restricción para el aspecto concepto. Las restricciones para este aspecto se sobrescriben.

ConstraintSet.set_entity_identifier_aspect(identifier, scheme)
Establece una restricción para el aspecto identificador de identidad. Las restricciones para este aspecto se sobrescriben.

ConstraintSet.set_period_aspect_instant(instant)
Establece una restricción para el aspecto período. Las restricciones para este aspecto se sobrescriben.

ConstraintSet.set_period_aspect_duration(start, end)
Establece una restricción para el aspecto duration. Las restricciones para este aspecto se sobrescriben.

ConstraintSet.set_period_aspect_forever()
Establece una restricción para el aspecto forever. Any existing constraint for this aspect will be overwritten.

ConstraintSet.set_segment_aspect(elements)
Establece una restricción para el aspecto segmento no XDT. Las restricciones para este aspecto se sobrescriben.

ConstraintSet.set_scenario_aspect(elements)
Establece una restricción para el aspecto escenario no XDT. Las restricciones para este aspecto se sobrescriben.

ConstraintSet.set_unit_aspect(nominators, denominators=None)
Establece una restricción para el aspecto unidad Las restricciones para este aspecto se sobrescriben.

ConstraintSet.dimension_aspect(dimension)
Devuelve un objeto `xbrl.ExplicitDimensionAspectValue` o `xbrl.TypedDimensionAspectValue` para el aspecto dimensión dado o `None` si no se especificó ninguna restricción.

ConstraintSet.set_dimension_aspect(dimensionAspectValue)
Establece una restricción para el aspecto dimensión. Las restricciones para este aspecto se sobrescriben.

ConstraintSet.set_explicit_dimension_aspect(dimension, domainMember)
Establece una restricción para el aspecto dimensión. Las restricciones para este aspecto se

 © 2014 Altova GmbH

sobrescriben.

`ConstraintSet.set_typed_dimension_aspect(dimension, element)`

Establece una restricción para el aspecto dimensión. Las restricciones para este aspecto se sobrescriben.

5.9.5 xbrl.Context

clase **`xbrl.Context`**
La clase `Context` representa un contexto XBRL del documento de instancia XBRL.

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

`Context.element`
Devuelve un objeto `xml.Element` que representa el elemento de información de elemento XML del contexto XBRL.

`Context.id`
Devuelve una cadena con el valor del atributo `id` del contexto XBRL.

`Context.period`
Devuelve un objeto `xbrl.Period` que representa la parte de período del contexto XBRL.

`Context.entity`
Devuelve un objeto `xbrl.Entity` que representa la parte de entidad del contexto XBRL.

`Context.scenario_element`
Devuelve un objeto `xml.Element` que representa el elemento de información de elemento XML `scenario` del contexto XBRL.

 © 2014 Altova GmbH

5.9.6 xbrl.DefinitionNodeResource

clase **xbrl.DefinitionNodeResource(xbrl.Resource)**
La clase DefinitionNodeResource representa un recurso definitionNode en la base de enlaces de tabla (http://www.xbrl.org/specification/table-linkbase/pr-2013-12-18/table-linkbase-pr-2013-12-18.html#sec-definition-nodes).

5.9.7 xbrl.DTS

clase **xbrl.DTS**
La clase DTS representa una colección de taxonomías XBRL por descubrir (DTS).

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

DTS.dimensions
Devuelve un iterador que recorre los objetos Concept que representan todos los conceptos XBRL de la DTS que están en el grupo de sustitución de `xbrldt:dimensionItem`.

DTS.hypercubes
Devuelve un iterador que recorre los objetos Concept que representan todos los conceptos XBRL de la DTS que están en el grupo de sustitución de `xbrldt:hypercubeItem`.

DTS.items
Devuelve un iterador que recorre los objetos Concept que representan todos los conceptos XBRL de la DTS que están en el grupo de sustitución de `xbrli:item`.

DTS.linkbases
Devuelve una lista de objetos `xml.Document` que representan todas las bases de enlaces XBRL de la DTS.

DTS.schema
Devuelve un objeto `xsd.Schema` que representa el componente de XML Schema de la DTS subyacente.

DTS.schemas
Devuelve una lista de objetos `xml.Document` que representan todos los esquemas de taxonomía XBRL de la DTS.

DTS.tables
Devuelve un objeto generador de objetos `xbrl.TableResource` que representan todos los recursos de tabla de la base de enlaces de tabla.

DTS.tuples
Devuelve un iterador que recorre los objetos Concept que representan todos los conceptos XBRL de la DTS que están en el grupo de sustitución de `xbrli:tuple`.

La clase DTS ofrece estos **métodos de instancia**:

DTS.resolve_concept(qname)
Devuelve un objeto Concept que representa el concepto XBRL con el nombre completo XML dado (`qname`) en la DTS. Si ese concepto no existe, este método devuelve `None`.

DTS.resolve_table(identifier)
Devuelve una lista de objetos `xbrl.TableResource` que coinciden con el identificador. El identificador puede ser el valor del atributo `id` o una URL XPointer que apunte al recurso de tabla.

 © 2014 Altova GmbH

5.9.8 xbrl.Entity

clase **`xbrl.Entity`**
La clase `Entity` representa la parte de entidad del contexto XBRL del documento de instancia.

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

`Entity.element`
Devuelve un objeto `xml.Element` que representa el elemento de información de elemento XML secundario `entity` del contexto XBRL.

`Entity.identifier`
Devuelve un objeto `xbrl.EntityIdentifier` que representa la parte de identificador de entidad del contexto XBRL.

`Entity.segment_element`
Devuelve un objeto `xml.Element` que representa el elemento de información de elemento XML secundario `segment` del contexto XBRL.

5.9.9 xbrl.EntityIdentifier

clase **xbrl.EntityIdentifier**
La clase EntityIdentifier representa la parte de identificador de entidad de un contexto XBRL del documento de instancia.

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

EntityIdentifier.aspect_value
Devuelve un objeto xbrl.EntityIdentifierAspectValue que representa el valor del aspecto identificador de entidad.

EntityIdentifier.element
Devuelve un objeto xml.Element que representa el elemento de información de elemento XML secundario identifier del contexto XBRL.

EntityIdentifier.value
Devuelve una cadena con el valor de texto del elemento de información de elemento XML secundario identifier del contexto XBRL.

EntityIdentifier.scheme
Devuelve una cadena con el valor de texto del atributo scheme del elemento de información de elemento XML secundario identifier.

 © 2014 Altova GmbH

5.9.10 xbrl.EntityIdentifierAspectValue

clase **`xbrl.EntityIdentifierAspectValue`**
La clase `EntityIdentifierAspectValue` representa un valor para el aspecto identificador de entidad. Use `xbrl.ConstraintSet` para crear objetos nuevos de esta clase.

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

`EntityIdentifierAspectValue.identifier`
Devuelve el identificador como cadena de texto.

`EntityIdentifierAspectValue.scheme`
Devuelve el modelo del identificador como cadena de texto.

5.9.11 xbrl.Error

clase **`xbrl.Error(Exception)`**
La clase es la clase base para los errores relacionados con XBRL.

 © 2014 Altova GmbH

5.9.12 xbrl.ExplicitDimensionAspectValue

clase **xbrl.ExplicitDimensionAspectValue**
La clase ExplicitDimensionAspectValue representa un valor para el aspecto dimensión. Use xbrl.ConstraintSet para crear objetos nuevos de esta clase.

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

ExplicitDimensionAspectValue.is_explicit
Devuelve `true`.

ExplicitDimensionAspectValue.is_typed
Devuelve `false`.

ExplicitDimensionAspectValue.dimension
Devuelve un objeto xbrl.Concept que representa la dimensión.

ExplicitDimensionAspectValue.value
Devuelve un objeto xbrl.Concept que representa el valor del miembro de dominio de la dimensión. El valor del aspecto dimensión ausente se denota con `None`.

5.9.13 xbrl.Fact

clase **xbrl.Fact**
La clase Fact representa un hecho de un documento de instancia XBRL.

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

Fact.element
Devuelve un objeto xml.Element que representa el elemento de información element del hecho XBRL.

Fact.qname
Devuelve un objeto xml.QName que representa el nombre XML completo del hecho XBRL.

Fact.concept
Devuelve un objeto Concept que representa el concepto XBRL asociado a este hecho XBRL.

Fact.context
Devuelve un objeto Context que representa el contexto XBRL asociado a este hecho XBRL.

Fact.unit
Devuelve un objeto Unit que representa la unidad XBRL asociada a este hecho XBRL.

Fact.nil
Devuelve True si el elemento de información de elemento del hecho XBRL tiene definido el atributo xsi:nil con valor True.

Fact.decimals
Devuelve una cadena que contiene el valor del atributo decimals del elemento de información element del hecho XBRL. Si no se definió el atributo decimals, devuelve None.

Fact.precision
Devuelve una cadena que contiene el valor del atributo precision del elemento de información element del hecho XBRL. Si no se definió el atributo precision, devuelve None.

Fact.normalized_value
Devuelve el valor normalizado de esquema del hecho XBRL en forma de cadena.

Fact.effective_numeric_value
Devuelve un objeto decimal.Decimal que representa el valor numérico efectivo del hecho XBRL tras redondearlo con la información suministrada por el atributo decimals o precision. Si la definición de tipo del hecho no se deriva de una definición de tipo numérico XBRL integrado, el atributo devuelve NaN.

Fact.fraction_value
Devuelve un objeto Fraction que representa el valor fraccionario que denota el hecho XBRL. Si la definición de tipo del hecho no se deriva de la definición de tipo XBRL integrado xbrli:fractionItemType, el atributo devuelve None.

Fact.child_facts

 © 2014 Altova GmbH

Devuelve un objeto generador de objetos `Fact` que representa todos los hechos que son secundarios directos de este hecho.

`Fact.child_items`
Devuelve un objeto generador de objetos `Fact` que representa los hechos XBRL de tipo ítem que son secundarios directos de este hecho.

`Fact.child_tuples`
Devuelve un objeto generador de objetos `Fact` que representa los hechos XBRL de tipo tupla que son secundarios directos de este hecho.

`Fact.footnote_elements`
Devuelve un objeto generador de objetos `xml.Element` que representa todos los recursos de tipo nota al pie asociados a este hecho.

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

`Fact.is_item()`
Devuelve `True` si se trata de un hecho XBRL de tipo ítem.

`Fact.is_tuple()`
Devuelve `True` si se trata de un hecho XBRL de tipo tupla.

`Fact.location_aspect()`
Devuelve un objeto `xml.Element` que representa el aspecto `location` de este hecho XBRL.

`Fact.concept_aspect()`
Devuelve un objeto`Concept` que representa el aspecto `concept` de este hecho XBRL.

`Fact.entity_identifier_aspect()`
Devuelve un objeto `EntityIdentifier` que representa el aspecto `entity identifier` de este hecho XBRL. Si el hecho XBRL no tiene el aspecto `entity identifier`, este atributo devuelve `None`.

`Fact.period_aspect()`
Devuelve un objeto `Period` que representa el aspecto `period` de este hecho XBRL. Si el hecho XBRL no tiene el aspecto `period`, este atributo devuelve `None`.

`Fact.unit_aspect()`
Devuelve un objeto `Unit` que representa el aspecto `unit` de este hecho XBRL. Si el hecho XBRL no tiene el aspecto `unit`, este atributo devuelve `None`.

`Fact.complete_segment_aspect()`
Devuelve un objeto `xml.Element` que representa el aspecto `complete segment` de este hecho XBRL. Si el hecho XBRL no tiene el aspecto `complete segment`, este atributo devuelve `None`.

`Fact.complete_scenario_aspect()`
Devuelve un objeto `xml.Element` que representa el aspecto `complete scenario` de este hecho XBRL. Si el hecho XBRL no tiene el aspecto `complete scenario`, este atributo devuelve `None`.

`Fact.non_xdt_segment_aspect()`

Devuelve una lista de objetos `xml.Element` que representa el aspecto `non-XDT segment` de este hecho XBRL. Si el hecho XBRL no tiene el aspecto `non-XDT segment`, este atributo devuelve `None`.

`Fact.non_xdt_scenario_aspect()`

Devuelve una lista de objetos `xml.Element` que representa el aspecto `non-XDT scenario` de este hecho XBRL. Si el hecho XBRL no tiene el aspecto `non-XDT scenario`, este atributo devuelve `None`.

`Fact.explicit_dimension_aspect(qname)`

Devuelve un objeto `xml.QName` que representa el aspecto dado `dimension` de **`QName`** de este hecho XBRL (el QName del miembro de dominio). Si el hecho XBRL no tiene ese aspecto `dimension`, este atributo devuelve `None`. El argumento **`QName`** debe ser un objeto de clase `xml.QName`.

`Fact.typed_dimension_aspect(qname)`

Devuelve un objeto `xml.Element` que representa el aspecto dado `dimension` de **`QName`** de este hecho XBRL (que es el elemento de dominio con tipo). Si el hecho XBRL no tiene ese aspecto `dimension`, este atributo devuelve `None`. El argumento **`QName`** debe ser un objeto de clase `xml.QName`.

 © 2014 Altova GmbH

5.9.14 xbrl.FactSet

clase **`xbrl.FactSet`**
La clase `FactSet` representa un conjunto de hechos de instancia XBRL.

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

`FactSet.copy()`
Devuelve un objeto `xbrl.FactSet` que es una copia independiente del conjunto de hechos actual.

`FactSet.merge(factSet)`
Combina los hechos representados por este objeto con el `factSet` dado (equivalente a una unión de conjuntos).

`FactSet.intersect(factSet)`
Intersecciona los hechos representados por este objeto con el `factSet` dado (equivalente a una intersección de conjuntos).

`FactSet.filter(constraintSet)`
Filtra los hechos que no coinciden con las restricciones especificadas en el `constraintSet`.

5.9.15 xbrl.Fraction

clase **`xbrl.Fraction(numerador, denominador)`**
Un objeto `Fraction` representa el valor fraccionario de un hecho XBRL de tipo `fractionItemType`. Construye un objeto de clase `Fraction`. Todos los argumentos son necesarios

---

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

`Fraction.numerator`
Devuelve un objeto `decimal.Decimal` que representa la parte numerador del valor fraccionario.

`Fraction.denominator`
Devuelve un objeto `decimal.Decimal` que representa la parte denominador del valor fraccionario.

 © 2014 Altova GmbH

5.9.16 xbrl.Instance

clase **xbrl.Instance**
Un objeto Instance representa un documento de instancia XBRL.

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

Instance.filename
Devuelve una cadena con el URL del documento de instancia XBRL.

Instance.document
Devuelve un objeto xml.Document que representa el elemento de información document del documento de instancia XBRL.

Instance.psvi
Devuelve un objeto xsd.PSVI que representa la información de esquema XML PSVI.

Instance.dts
Devuelve un objeto DTS que representa la colección de taxonomías XBRL por descubrir (DTS).

Instance.contexts
Devuelve un objeto generador de objetos Context que representa todos los contextos XBRL presentes en el documento de instancia.

Instance.units
Devuelve un objeto generador de objetos Unit que representa todas las unidades XBRL presentes en el documento de instancia.

Instance.facts
Devuelve un objeto generador de objetos Fact que representa todos los hechos XBRL presentes en el documento de instancia XBRL. También incluye los hechos que son elementos secundarios de otras tuplas.

Instance.non_nil_facts
Devuelve un objeto generador de objetos Fact que representa todos los hechos XBRL que no son nil presentes en el documento de instancia XBRL. También incluye los hechos que son elementos secundarios de otras tuplas.

Instance.items
Devuelve un objeto generador de objetos Fact que representa todos los hechos XBRL de nivel superior presentes en el documento de instancia XBRL. No se incluyen los hechos que son secundarios de otras tuplas.

Instance.select_facts(constraintSet)
Devuelve un objeto xbrl.FactSet que representa los hechos que coinciden con las restricciones del constraintSet.

Instance.tuples
Devuelve un objeto generador de objetos Fact que representa todos los hechos de tupla XBRL de nivel superior del documento de instancia. No se incluyen los hechos que son secundarios de otras tuplas.

© 2014 Altova GmbH

5.9.17 xbrl.LabelResource

clase **xbrl.LabelResource(xbrl.Resource)**
La clase LabelResource representa un recurso de etiqueta.

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

LabelResource.lang
Devuelve el valor del atributo xml:lang como cadena de texto.

5.9.18 xbrl.LayoutCell

clase **xbrl.LayoutCell**
La clase LayoutCell representa una celda en la tabla generada.

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

LayoutCell.row
Devuelve la coordinada Y de la celda de la tabla.

LayoutCell.col
Devuelve la coordinada X de la celda de la tabla.

LayoutCell.row_span
Devuelve el intervalo de celdas de la tabla en la dirección vertical.

LayoutCell.col_span
Devuelve el intervalo de celdas de la tabla en la dirección horizontal.

LayoutCell.is_span_start
Devuelve `True` si esta celda es la celda superior izquierda de la zona ocupada. También devuelve `True` si la celda de la tabla no tiene un intervalo.

LayoutCell.is_header
Devuelve `True` si esta celda representa una celda de encabezado.

LayoutCell.is_data
Devuelve `True` si esta tabla representa una celda de datos.

 © 2014 Altova GmbH

5.9.19 xbrl.LayoutDataCell

clase **xbrl.LayoutDataCell(xbrl.LayoutCell)**
La clase LayoutDataCell representa una celda de datos en la tabla generada.

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

LayoutDataCell.facts
Devuelve un objeto xbrl.FactSet que representa los hechos que coinciden con las restricciones de la celda.

5.9.20 xbrl.LayoutHeaderCell

clase **xbrl.LayoutHeaderCell(xbrl.LayoutCell)**
La clase LayoutHeaderCell representa una celda de encabezado en la tabla generada.

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

LayoutHeaderCell.definition_node
Devuelve un objeto xbrl.DefinitionNodeResource que representa el recurso de nodo de definición original en la base de enlaces de tabla.

LayoutHeaderCell.preferred_label_role
Para las celdas de encabezado generadas por un examen del árbol de la base de enlaces de presentación, se devuelve el valor preferredLabel de la relación que da lugar a un concepto determinado. De lo contrario se devuelve None.

LayoutHeaderCell.constraint_sets
Devuelve un objeto generador de etiquetas para las asignaciones del conjunto de restricciones. Puede tener acceso al conjunto de restricciones sin etiquetas usando None como clave.

LayoutHeaderCell.tag_selectors
Devuelve un objeto generador de selectores de etiqueta (especificados directamente o heredados).

LayoutHeaderCell.is_rollup
Devuelve True si la celda de encabezado representa un nodo de acumulación (http://www.xbrl.org/specification/table-linkbase/pr-2013-12-18/table-linkbase-pr-2013-12-18.html#sec-roll-up-nodes).

La clase LayoutHeaderCell ofrece estos **métodos de instancia**:

LayoutHeaderCell.constraint_set(tag=None)
Devuelve un objeto xbrl.ConstraintSet para la etiqueta dada o None si no se encontró ningún conjunto de restricciones con la etiqueta dada. Si la etiqueta es None, se devuelve el conjunto de restricciones sin etiquetar.

 © 2014 Altova GmbH

5.9.21 xbrl.LayoutRow

clase **xbrl.LayoutRow**
La clase LayoutRow representa una fila de la tabla generada.

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

LayoutRow.index
Devuelve las coordinadas de la fila de la tabla.

LayoutRow.is_header_row
Devuelve True si esta fila contiene celdas de encabezado solamente.

LayoutRow.cells
Devuelve un objeto generador de objetos xbrl.LayoutHeaderCell y xbrl.LayoutDataCell que representan las celdas de la fila de la tabla (celdas de encabezado y de datos).

5.9.22 xbrl.LayoutTable

clase **xbrl.LayoutTable**
La clase LayoutTable representa la tabla después de aplicar el proceso de resolución y diseño.

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

LayoutTable.tables
Devuelve un objeto xbrl.TableResource que representa el recurso de tabla original de la base de enlaces de tabla.

LayoutTable.rows
Devuelve un objeto generador de objetos xbrl.LayoutRow que representa las filas de la tabla (tanto filas de datos como de encabezado).

LayoutTable.width
Devuelve el número total de columnas de la tabla (columnas de datos y de encabezado).

LayoutTable.height
Devuelve el número total de filas de la tabla (filas de datos y de encabezado).

LayoutTable.header_width
Devuelve el número de columnas de encabezado.

LayoutTable.header_height
Devuelve el número de filas de encabezado.

LayoutTable.data_width
Devuelve el número de columnas de datos.

LayoutTable.data_height
Devuelve el número de filas de datos.

La clase LayoutTable ofrece estos **métodos de instancia**:

LayoutTable.cell(row, col)
Devuelve un objeto xbrl.LayoutCell que representa la celda de tabla de la fila especificada por `row` y la columna especificada por `col`.

 © 2014 Altova GmbH

5.9.23 xbrl.LayoutTableSet

clase **`xbrl.LayoutTableSet`**
La clase `LayoutTableSet` representa el conjunto de tablas después de aplicar el proceso de resolución y diseño.

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

`LayoutTableSet.tables`
Devuelve un objeto generador de objetos `xbrl.LayoutTable`.

5.9.24 xbrl.Period

clase **xbrl.Period**
La clase Period representa la parte de período de un contexto XBRL del documento de instancia.

La clase Period ofrece estas **constantes**:

Period.INSTANT
Denota que el período contiene un elemento de información XML secundario `instant`.

Period.DURATION
Denota que el período contiene los elementos de información XML secundarios `startDate` y `endDate`.

Period.FOREVER
Denota que el período contiene un elemento de información XML secundario `forever`.

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

Period.aspect_value
Devuelve un objeto que representa el valor del aspecto período.

Period.element
Devuelve un objeto `xml.Element` que representa el elemento de información XML secundario `period` del contexto XBRL.

Period.period_type
Devuelve INSTANT, DURATION o FOREVER dependiendo del elemento de información XML secundario del período.

Period.instant
Devuelve una cadena con el valor de texto del elemento de información de elemento XML secundario `instant`. Si el período no tiene un elemento de información de elemento XML secundario `instant`, devuelve `None`.

Period.start_date
Devuelve una cadena con el valor de texto del elemento de información de elemento XML secundario `startDate`. Si el período no tiene un elemento de información de elemento XML secundario `startDate`, devuelve `None`.

Period.end_date
Devuelve una cadena con el valor de texto del elemento de información de elemento XML secundario `endDate`. Si el período no tiene un elemento de información de elemento XML secundario `endDate`, devuelve `None`.

Period.effective_instant
Devuelve un objeto `datetime.datetime` con la fecha y hora calculada a partir del valor del elemento de información de elemento XML secundario `instant`. Si el período no tiene un elemento de información de elemento XML secundario `instant`, devuelve `None`.

 © 2014 Altova GmbH

`Period.effective_start_date`

Devuelve un objeto `datetime.datetime` con la fecha y hora calculada a partir del valor del elemento de información de elemento XML secundario `startDate`. Si el período no tiene un elemento de información de elemento XML secundario `startDate`, devuelve `None`.

`Period.effective_end_date`

Devuelve un objeto `datetime.datetime` con la fecha y hora calculada a partir del valor del elemento de información de elemento XML secundario `endDate`. Si el período no tiene un elemento de información de elemento XML secundario `endDate`, devuelve `None`.

5.9.25 xbrl.PeriodAspectValue

clase **xbrl.PeriodAspectValue**
La clase PeriodAspectValue representa un valor para el aspecto período. Use xbrl.ConstraintSet para crear objetos nuevos de esta clase.

La clase PeriodAspectValue ofrece estas **constantes**:

PeriodAspectValue.INSTANT
Denota que el período representa un punto en el tiempo.

PeriodAspectValue.DURATION
Denota que el período representa una duración con un punto de inicio y un punto final.

PeriodAspectValue.FOREVER
Detona que el período representa una duración sin un punto de inicio ni punto final.

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

PeriodAspectValue.period_type
Devuelve INSTANT, DURATION, or FOREVER.

PeriodAspectValue.instant
Si period_type es INSTANT, devuelve un objeto datetime.datetime que representa un solo punto en el tiempo. De lo contrario, devuelve None.

PeriodAspectValue.start
Si period_type es DURATION, devuelve un objeto datetime.datetime que representa el punto de inicio de la duración. De lo contrario, devuelve None.

PeriodAspectValue.end
Si period_type es DURATION, devuelve un objeto datetime.datetime que representa el punto final de la duración. De lo contrario, devuelve None.

 © 2014 Altova GmbH

5.9.26 xbrl.ReferenceResource

clase **xbrl.ReferenceResource(xbrl.Resource)**
La clase ReferenceResource representa un recurso de referencia.

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

ReferenceResource.lang
Devuelve el valor del atributo xml:lang como cadena de texto.

5.9.27 xbrl.Resource

clase **xbrl.Resource**
La clase Resource representa un elemento recurso XLink dentro de un vínculo extendido.

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

Resource.element
Devuelve un objeto xml.Element que representa el elemento XML del vínculo extendido.

Resource.id
Devuelve el valor del atributo id como cadena de texto o devuelve None si el elemento XML no tiene el atributo id.

Resource.role
Devuelve el valor del atributo xlink:role como cadena de texto o devuelve None si el elemento XML no tiene el atributo xlink:role.

Resource.labels
Devuelve un objeto generador de objetos xbrl.LabelResource por cada etiqueta genérica que tenga una relación etiqueta-elemento (http://xbrl.org/arcrole/2008/element-label) con este recurso.

Resource.references
Devuelve un objeto generador de objetos xbrl.ReferenceResource por cada etiqueta genérica que tenga una relación referencia-elemento (http://xbrl.org/arcrole/2008/element-reference) con este recurso.

 © 2014 Altova GmbH

5.9.28 xbrl.ScenarioAspectValue

clase **xbrl.ScenarioAspectValue**

La clase ScenarioAspectValue representa un valor para aspecto escenario no XDT. Use xbrl.ConstraintSet para crear objetos nuevos de esta clase.

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

ScenarioAspectValue.elements

Devuelve un objeto generador de objetos xml.Element que representa los elementos secundarios no XDT del elemento XML <scenario>.

5.9.29 xbrl.SegmentAspectValue

clase **xbrl.SegmentAspectValue**

La clase SegmentAspectValue representa un valor para el aspecto segmento no XDT. Use xbrl.ConstraintSet para crear objetos nuevos de esta clase.

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

SegmentAspectValue.elements

Devuelve un objeto generador de objetos xml.Element que representa los elementos secundarios no XDT del elemento XML .

 © 2014 Altova GmbH

5.9.30 xbrl.TableError

clase **xbrl.TableError(xbrl.Error)**

La clase xbrl.TableError representa un error que se emita durante el proceso de resolución o de diseño de tablas.

5.9.31 xbrl.TableResource

clase **xbrl.TableResource(xbrl.Resource)**
La clase TableResource representa un recurso de tabla en la base de enlaces de tabla (http://www.xbrl.org/specification/table-linkbase/pr-2013-12-18/table-linkbase-pr-2013-12-18.html#sec-tables).

La clase TableResource ofrece este **método de instancia**:

TableResource.layout(instance)
Devuelve un objeto xbrl.LayoutTableSet que representa el conjunto de tablas generado. Si ocurre un error durante el proceso de resolución o de diseño, se emite xbrl.TableError.

 © 2014 Altova GmbH

5.9.32 xbrl.TypedDimensionAspectValue

clase **xbrl.TypedDimensionAspectValue**
La clase TypedDimensionAspectValue representa un valor para el aspecto dimensión. Use xbrl.ConstraintSet para crear objetos nuevos de esta clase.

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

TypedDimensionAspectValue.is_explicit
Devuelve `true`.

TypedDimensionAspectValue.is_typed
Devuelve `false`.

TypedDimensionAspectValue.dimension
Devuelve un objeto xbrl.Concept que representa la dimensión.

TypedDimensionAspectValue.value
Devuelve un objeto xml.Element que representa el valor del dominio con tipo de la dimensión. El valor del aspecto dimensión ausente se denota con `None`.

5.9.33 xbrl.Unit

clase **xbrl.Unit**
La clase Unit representa una unidad XBRL del documento de instancia.

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

Unit.aspect_value
Devuelve un objeto xbrl.UnitAspectValue que representa el valor del aspecto unidad.

Unit.element
Devuelve un objeto xml.Element que representa el elemento de información element de la unidad XBRL.

Unit.id
Devuelve una cadena con el valor del atributo id de la unidad XBRL.

Unit.numerators
Devuelve una lista de objetos xml.QName que representa los QName del numerador de la unidad XBRL.

Unit.denominators
Devuelve una lista de objetos xml.QName que representa los QName del denominador de la unidad XBRL.

 © 2014 Altova GmbH

5.9.34 xbrl.UnitAspectValue

clase **xbrl.UnitAspectValue**

La clase UnitAspectValue representa un valor para el aspecto unidad. Use xbrl.ConstraintSet para crear objetos nuevos de esta clase.

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

UnitAspectValue.numerators

Devuelve una lista de objetos xml.QName que representa los QName del numerador de la unidad XBRL.

UnitAspectValue.denominators

Devuelve una lista de objetos xml.QName que representa los QName del denominador de la unidad XBRL.

Altova RaptorXML+XBRL Server 2014

Interfaz Java

6 Interfaz Java

A la API de RaptorXML también se puede acceder desde código Java. Para ello es necesario que las bibliotecas que aparecen más abajo residan en el parámetro classpath. Esta bibliotecas se instalan en la carpeta `bin` de la carpeta de instalación.

- `RaptorXMLServer.jar`: es la biblioteca que se comunica con el servidor de RaptorXML mediante solicitudes HTTP.
- `RaptorXMLServer_JavaDoc.zip`: es un archivo Javadoc que contiene la documentación de ayuda para la API de Java.

Nota: para poder usar la API de Java, el archivo Jar debe estar en el parámetro classpath. Si quiere, puede copiar el archivo Jar en la ubicación que mejor le venga a su proyecto en lugar de hacerle referencia desde la carpeta de instalación.

Resumen sobre la interfaz Java
La API de Java viene empaquetada en el paquete `com.altova.raptorxml`. La clase RaptorXML ofrece un método de punto de entrada llamado `getFactory()`, que devuelve objetos `RaptorXMLFactory`. De modo que se puede crear una instancia de `RaptorXMLFactory` con la llamada `RaptorXML.getFactory()`.

La interfaz `RaptorXMLFactory` ofrece métodos para obtener objetos del motor para realizar tareas de validación y procesamiento (como transformaciones XSLT).

Nota: el método `getFactory` devuelve el objeto de fábrica correspondiente dependiendo de la edición de RaptorXML que esté instalada.

La interfaz pública de `RaptorXMLFactory` se describe en este fragmento:

```
public interface RaptorXMLFactory
{
   public XMLValidator getXMLValidator();
   public XBRL getXBRL();
   public XQuery getXQuery();
   public XSLT getXSLT();
   public void setServerName(String name) throws RaptorXMLException;
   public void setServerFile(String file) throws RaptorXMLException;
   public void setServerPort(int port) throws RaptorXMLException;
   public void setGlobalCatalog(String catalog);
   public void setUserCatalog(String catalog);
   public void setGlobalResourcesFile(String file);
   public void setGlobalResourceConfig(String config);
   public void setErrorFormat(ENUMErrorFormat format);
   public void setErrorLimit(int limit);
   public void setReportOptionalWarnings(boolean report);
```

 © 2014 Altova GmbH

```
}
```

Para más información consulte la descripción de `RaptorXMLFactory` y de la correspondiente interfaz Java. También puede consultar un ejemplo de proyecto Java.

6.1 Ejemplo de proyecto Java

El fragmento de código Java que aparece más adelante muestra cómo acceder a las funciones básicas. El código tiene varias partes:

- Busca la carpeta de ejemplos y crea una instancia de objeto COM de RaptorXML
- Valida un archivo XML
- Realiza una transformación XSLT y devuelve el resultado en forma de cadena de texto
- Procesa un documento XQuery y devuelve el resultado en forma de cadena de texto
- Ejecuta el proyecto

Estas funciones básicas se incluyen en los archivos de la carpeta `examples/API` de la carpeta de aplicación de RaptorXML+XBRL Server.

```
public class RunRaptorXML
{
    // Buscar muestras instaladas con el producto
    // (está dos niveles más arriba de examples/API/Java)
    // NOTA: quizás sea necesario cambiar esta ruta de acceso
       static final String strExamplesFolder = System.getProperty("user.dir") +
"/../../" ;

       static com.altova.raptorxml.RaptorXMLFactory rxml;
```

```
       static void ValidateXML() throws com.altova.raptorxml.RaptorXMLException
       {
           com.altova.raptorxml.XMLValidator xmlValidator =
rxml.getXMLValidator();
           System.out.println("RaptorXML Java - XML validation");
           xmlValidator.setInputXMLFromText( "<!DOCTYPE root [ <!ELEMENT root
(#PCDATA)> ]> <root>simple input document</root>" );
              if( xmlValidator.isWellFormed() )
                  System.out.println( "The input string is well-formed" );
              else
                  System.out.println( "Input string is not well-formed: " +
xmlValidator.getLastErrorMessage() );

              if( xmlValidator.isValid() )
                  System.out.println( "The input string is valid" );
              else
                  System.out.println( "Input string is not valid: " +
xmlValidator.getLastErrorMessage() );
       }
```

```
       static void RunXSLT() throws com.altova.raptorxml.RaptorXMLException
       {
```

```
            System.out.println("RaptorXML Java - XSL Transformation");
            com.altova.raptorxml.XSLT xsltEngine = rxml.getXSLT();
            xsltEngine.setInputXMLFileName( strExamplesFolder + "simple.xml" );
            xsltEngine.setXSLFileName( strExamplesFolder + "transform.xsl" );
            String result = xsltEngine.executeAndGetResultAsString();
            if( result == null )
                System.out.println( "Transformation failed: " +
xsltEngine.getLastErrorMessage() );
            else
                System.out.println( "Result is " + result );
        }
```

```
        static void RunXQuery() throws com.altova.raptorxml.RaptorXMLException
        {
            System.out.println("RaptorXML Java - XQuery execution");
            com.altova.raptorxml.XQuery xqEngine = rxml.getXQuery();
            xqEngine.setInputXMLFileName( strExamplesFolder + "simple.xml" );
            xqEngine.setXQueryFileName( strExamplesFolder + "CopyInput.xq" );
            System result = xqEngine.executeAndGetResultAsString();
            if( result == null )
                System.out.println( "Execution failed: " +
xqEngine.getLastErrorMessage() );
            else
                System.out.println( "Result is " + result );
        }
```

```
        public static void main(String[] args)
        {
            try
            {
                rxml = com.altova.raptorxml.RaptorXML.getFactory();
                rxml.setErrorLimit( 3 );

                ValidateXML();
                RunXSLT();
                RunXQuery();
            }

            catch( com.altova.raptorxml.RaptorXMLException e )
            {
                e.printStackTrace();
            }

        }

}
```

6.2 Interfaces de RaptorXML para Java

A continuación resumimos las interfaces Java de la API de RaptorXML. Para más información consulte los apartados correspondientes.

- `RaptorXMLFactory`
  Crea una instancia de objeto COM de RaptorXML nueva por medio de una llamada nativa y ofrece acceso a los motores de RaptorXML.
- `XMLValidator`
  Interfaz para el motor de validación XML.
- `XSLT`
  Interfaz para los motores XSLT.
- `XQuery`
  Interfaz para los motores XQuery.
- `XBRL`
  Interfaz para el motor XBRL.
- `RaptorXMLException`
  Interfaz para el método `RaptorXMLException`.

 © 2014 Altova GmbH

6.2.1 RaptorXMLFactory

`public interface` **`RaptorXMLFactory`**

Descripción

Use `RaptorXMLFactory()` para crear una nueva instancia de objeto COM de RaptorXML. Esto ofrece acceso a los motores de RaptorXML. `RaptorXMLFactory` y el objeto COM de RaptorXML tienen una relación de iguales. Es decir, en adelante las llamadas a la función `get<NOMBREMOTOR>()` devolverán interfaces para la misma instancia del motor.

Más abajo describimos primero los métodos de la interfaz `RaptorXMLFactory` y después sus enumeraciones.

Métodos

Los métodos de la clase se describen por orden alfabético. En la tabla se organizan por grupos para facilitar la consulta.

Motores	Errores, advertencias
getXBRL	setErrorFormat
getXMLValidator	setErrorLimit
getXQuery	setReportOptionalWarnings
getXSLT	

Catálogos	Recursos globales	Servidor HTTP
setGlobalCatalog	setGlobalResourceConfig	setServerFile
setUserCatalog	setGlobalResourcesFile	setServerName
		setServerPort

Información sobre el producto	
getProductName	Is64Bit
getProductNameAndVersion	getAPIMajorVersion
getMajorVersion	getAPIMinorVersion
getMinorVersion	getAPIServicePackVersion
getServicePackVersion	

Subir | **Métodos** | **Enumeraciones**

getAPIMajorVersion

`public int` **`getAPIMajorVersion()`**

Devuelve la versión principal de la API en forma de entero. La versión principal de la API puede ser distinta a la versión principal del producto si la API está conectada a otro servidor.
Devuelve: un entero que es la versión principal de la API.

Subir | **Métodos** | **Enumeraciones**

getAPIMinorVersion

`public int` **`getAPIMinorVersion()`**

Devuelve la versión secundaria de la API en forma de entero. La versión secundaria de la API puede ser distinta a la versión secundaria del producto si la API está conectada a otro servidor.
Devuelve: un entero que es la versión secundaria de la API.

Subir | **Métodos** | **Enumeraciones**

getAPIServicePackVersion

`public int` **`getAPIServicePackVersion()`**

Devuelve la versión del service pack de la API en forma de entero. La versión del service pack de la API puede ser distinta a la versión del service pack del producto si la API está conectada a otro servidor.
Devuelve: un entero que es la versión del service pack de la API.

Subir | **Métodos** | **Enumeraciones**

getMajorVersion

`public int` **`getMajorVersion()`**

Devuelve la versión principal del producto en forma de entero. Ejemplo: para `Altova RaptorXML +XBRL Server 2014r2sp1(x64)`, devuelve `16` (la diferencia entre la versión principal (`2014`) y el año inicial `1998`). Emite la excepción `RaptorXMLException` en caso de error.
Devuelve: un entero que es la versión principal del producto.

Subir | **Métodos** | **Enumeraciones**

getMinorVersion

`public int` **`getMinorVersion()`**

Devuelve la versión secundaria del producto en forma de entero. Ejemplo: para `Altova RaptorXML+XBRL Server 2014r2sp1(x64)`, devuelve `2` (tomado del número de versión secundaria `r2`). Emite la excepción `RaptorXMLException` en caso de error.
Devuelve: un entero que es la versión secundaria del producto.

Subir | **Métodos** | **Enumeraciones**

 © 2014 Altova GmbH

getProductName
`public string getProductName()`
Devuelve el nombre del producto en forma de cadena de texto. Ejemplo: para `Altova RaptorXML +XBRL Server 2014r2sp1(x64)`, devuelve `Altova RaptorXML+XBRL Server`. Emite la excepción `RaptorXMLException` en caso de error.
Devuelve: una cadena de texto que es el nombre del producto.

Subir | Métodos | Enumeraciones

getProductNameAndVersion
`public string getProductNameAndVersion()`
Devuelve la versión del service pack del producto en forma de entero. Ejemplo: para `Altova RaptorXML+XBRL Server 2014r2sp1(x64)`, devuelve `Altova RaptorXML+XBRL Server 2014r2sp1(x64)`. Emite la excepción `RaptorXMLException` en caso de error.
Devuelve: una cadena de texto que es el nombre y la versión del producto.

Subir | Métodos | Enumeraciones

getServicePackVersion
`public int getServicePackVersion()`
Devuelve la versión del service pack del producto en forma de entero. Ejemplo: para `RaptorXML +XBRL Server 2014r2sp1(x64)`, devuelve `1` (del número de versión del service pack `sp1`). Emite la excepción `RaptorXMLException` en caso de error.
Devuelve: un entero que es la versión del service pack del producto.

Subir | Métodos | Enumeraciones

getXBRL
`public XBRL getXBRL`
Recupera el motor XBRL.
Devuelve: una instancia nueva de `XBRL` de este `RaptorXMLFactory`.

Subir | Métodos | Enumeraciones

getXMLValidator
`public XMLValidator getXMLValidator()`
Recupera el validador XML.
Devuelve: una instancia nueva de `XMLValidator` de este `RaptorXMLFactory`.

Subir | Métodos | Enumeraciones

getXQuery
`public XQuery getXQuery()`
Recupera el motor XQuery.
Devuelve: una instancia nueva de `XQuery` de este `RaptorXMLFactory`.

Subir | Métodos | Enumeraciones

getXSLT
`public XSLT getXSLT()`
Recupera el motor XSLT.
Devuelve: una instancia nueva de `XSLT` de este `RaptorXMLFactory`.

Subir | Métodos | Enumeraciones

is64Bit
`public boolean is64Bit()`
Comprueba si la aplicación es un ejecutable de 64 bits. Ejemplo: para `Altova RaptorXML+XBRL Server 2014r2sp1(x64)`, devuelve `true`. Emite la excepción `RaptorXMLException` en caso de error.
Devuelve: el booleano `true` si la aplicación es de 64 bits y `false` si no lo es.

Subir | Métodos | Enumeraciones

setErrorFormat
`public void setErrorFormat(ENUMErrorFormat format)`
Define el formato de los errores de RaptorXML como uno de los literales `ENUMErrorFormat` (Text, ShortXML, LongXML).
Parámetros: `format`: almacena el valor del literal `ENUMErrorFormat` seleccionado.

Subir | Métodos | Enumeraciones

setErrorLimit
`public void setErrorLimit(int limit)`
Define el límite de errores de validación de RaptorXML.
Parámetros: `limit`: es de tipo `int` y especifica el número de errores que se deben comunicar antes de detener la ejecución. Utilice `-1` para definir `limit` como ilimitado (es decir, para comunicar todos los errores). El valor predeterminado es `100`.

Subir | Métodos | Enumeraciones

 © 2014 Altova GmbH

setGlobalCatalog

`public void setGlobalCatalog(String catalog)`

Define la ubicación en forma de URL del archivo de catálogo principal (de punto de entrada).
Parámetros: `catalog`: la cadena suministrada debe ser una URL absoluta que dé la ubicación exacta del archivo de catálogo principal que debe utilizarse.

Subir | Métodos | Enumeraciones

setGlobalResourceConfig

`public void setGlobalResourceConfig(String config)`

Establece la configuración activa del recurso global.
Parámetros: `config`: es de tipo `String` y especifica el nombre de la configuración utilizada por el recurso global activo.

Subir | Métodos | Enumeraciones

setGlobalResourcesFile

`public void setGlobalResourcesFile(String file)`

Establece la ubicación en forma de URL del archivo XML de recursos globales.
Parámetros: `file`: la cadena suministrada debe ser una URL absoluta que dé la ubicación exacta del archivo XML de recursos globales.

Subir | Métodos | Enumeraciones

setReportOptionalWarnings

`public void setReportOptionalWarnings(boolean report)`

Habilita/deshabilita la notificación de advertencias. Un valor `true` habilita las advertencias; `false` las deshabilita.
Parámetros: `report`: toma el booleano `true` o `false`.

Subir | Métodos | Enumeraciones

setServerFile

`public void setServerFile(String file)`

Establece la ubicación del archivo de configuración del servidor HTTP relativa a la dirección del servidor HTTP. Emite la excepción `RaptorXMLException` en caso de error.
Parámetros: `file`: una cadena de texto que indica la dirección del archivo de configuración del servidor HTTP relativa a la dirección del servidor.

Subir | Métodos | Enumeraciones

setServerName

```
public void setServerName(String name)
```

Establece el nombre del servidor HTTP. Emite la excepción RaptorXMLException en caso de error.

Parámetros: name: una cadena que indica el nombre del servidor HTTP.

Subir | Métodos | Enumeraciones

setServerPort

```
public void setServerPort(int port)
```

Establece el puerto del servidor HTTP a través del cual se accede al servicio. El puerto debe ser fijo y conocido para que las solicitudes HTTP puedan apuntar correctamente al servicio. Emite la excepción RaptorXMLException en caso de error.

Parámetros: port: un entero que indica el puerto de acceso del servidor HTTP.

Subir | Métodos | Enumeraciones

setUserCatalog

```
public void setUserCatalog(String catalog)
```

Establece la ubicación en forma de URL del archivo de catálogo personal del usuario.

Parámetros catalog: la cadena indicada debe ser una URL absoluta que dé la ubicación exacta del archivo de catálogo personal que se debe usar.

Subir | Métodos | Enumeraciones

Enumeraciones
ENUMErrorFormat

ENUMErrorFormat

```
public enum ENUMErrorFormat {
        eFormatText
        eFormatShortXML
        eFormatLongXML }
```

ENUMErrorFormat puede tomar uno de estos literales de enumeración: eFormatText, eFormatShortXML, eFormatLongXML. Estos literales definen el formato de los mensajes de error (eLongXML aporta los mensajes de error más detallados). El literal predeterminado es eFormatText.

Utilizado por (Interfaz::Método):

RaptorXMLFactory setErrorFormat

 © 2014 Altova GmbH

Subir | Métodos | Enumeraciones

6.2.2 XMLValidator

```
public interface XMLValidator
```

Descripción
Valida el documento XML, esquema o DTD suministrados. La validación de documentos XML se puede hacer con DTD o esquemas XML internos o externos. También comprueba si los documentos tienen un formato XML correcto. A continuación describimos primero los métodos de la interfaz y después sus enumeraciones.

Métodos
Los métodos de la clase se describen por orden alfabético. En la tabla se organizan por grupos para facilitar la consulta.

Procesamiento	Archivos de entrada	XML Schema
isValid(ENUM type)	setInputXMLFileName	setSchemaImports
isValid	setInputXMLFromText	setSchemalocationHints
isWellFormed(ENUM type)	setInputXMLFileCollection	setSchemaMapping
isWellFormed	setInputXMLTextCollection	setXSDVersion
getLastErrorMessage	setSchemaFileName	
setAssessmentMode	setSchemaFromText	**XML**
setPythonScriptFile	setSchemaFileCollection	setEnableNamespaces
setStreaming	setSchemaTextCollection	setXincludeSupport
	setDTDFileName	setXMLValidationMode
	setDTDFromText	

Subir | Métodos | Enumeraciones

getLastErrorMessage

```
public String getLastErrorMessage()
```

Recupera el último mensaje de error del motor de validación XML.
Devuelve: una cadena que es el último mensaje de error del motor de validación XML.

Subir | Métodos | Enumeraciones

isValid

public boolean **isValid**(ENUMValidationType type)
Devuelve el resultado de validar el documento XML, esquema o DTD. El tipo de documento se indica con el parámetro type, que toma como valor un literal de ENUMValidationType. El resultado es true si el documento es válido y false si no lo es. Si ocurre un error, se emite una excepción RaptorXMLException. Use el método getLastErrorMessage para obtener más información.
Parámetros: type: un literal de ENUMValidationType que especifica si se valida un esquema XML, un DTD o un documento XML con un esquema XML o si se valida un documento XML con un documento DTD.
Devuelve: el booleano true si el documento es válido y false si no lo es.

Subir | **Métodos** | **Enumeraciones**

isValid
public boolean **isValid()**
Devuelve el resultado de validar el documento suministrado.
Devuelve: el booleano true si el documento es válido y false si no lo es.

Subir | **Métodos** | **Enumeraciones**

isWellFormed
public boolean **isWellFormed**(ENUMWellformedCheckType type)
Devuelve el resultado de la comprobación de formato XML del documento XML o DTD. El tipo de documento cuyo formato se comprueba viene dado por el parámetro type, que toma como valor un literal de ENUMWellformedCheckType. Si el formato es correcto el resultado es true y false si no lo es. Si ocurre un error, se emite una excepción RaptorXMLException. Use el método getLastErrorMessage para obtener más información.
Parámetros: type: un literal de ENUMWellformedCheckType que especifica si el documento cuyo formato se comprueba es un XML o un DTD.
Devuelve: el booleano true si el formato es correcto y false si no lo es.

Subir | **Métodos** | **Enumeraciones**

isWellFormed
public boolean **isWellFormed()**
Devuelve el resultado de la comprobación de formato XML del documento XML o DTD. Si el formato es correcto el resultado es true y false si no lo es.
Devuelve: el booleano true si el formato es correcto y false si no lo es.

Subir | **Métodos** | **Enumeraciones**

setAssessmentMode
public void **setAssessmentMode**(ENUMAssessmentMode mode)

Establece el modo de evaluación de la validación XML (`Strict/Lax`), que se define en el parámetro `mode`, que toma como valor un literal de `ENUMAssessmentMode`.
Parámetros: `mode`: un literal de `ENUMAssessmentMode` que especifica si la validación debe strict o lax o si se debe pasar por alto.

Subir | Métodos | Enumeraciones

setDTDFileName

```
public void setDTDFileName(String filePath)
```

Establece la ubicación en forma de URL del documento DTD que se debe usar para la validación.
Parámetros: `filePath`: cadena de texto que debe ser una URL absoluta que dé la ubicación exacta del archivo DTD que se debe usar.

Subir | Métodos | Enumeraciones

setDTDFromText

```
public void setDTDFromText(String dtdText)
```

Aporta el contenido del documento DTD en forma de texto.
Parámetros: `dtdText`: cadena de texto es el documento DTD que se debe usar para la validación.

Subir | Métodos | Enumeraciones

setEnableNamespaces

```
public void setEnableNamespaces(boolean enable)
```

Habilita el procesamiento preparado para espacios de nombres. Es útil si quiere buscar errores en la instancia XML derivados del uso incorrecto de espacios de nombres. El valor `true` habilita el procesamiento preparado para espacios de nombres; `false` lo deshabilita. El predeterminado es `false`.
Parámetros: `support`: toma el booleano `true` o `false`.

Subir | Métodos | Enumeraciones

setInputXMLFileCollection

```
public void setInputXMLFileCollection(Collection<?> fileCollection)
```

Aporta la colección de archivos XML que se usará como datos de entrada. Los archivos se identifican con su dirección URL.
Parámetros: `fileCollection`: colección de cadenas de texto, cada una de las cuales es la URL absoluta de un archivo XML de entrada.

Subir | Métodos | Enumeraciones

 © 2014 Altova GmbH

setInputXMLFileName

```
public void setInputXMLFileName(String filePath)
```

Establece la ubicación en forma de URL del documento XML que se debe validar.
Parámetros: `filePath`: cadena de texto que debe ser una URL absoluta que dé la ubicación exacta del archivo XML.

Subir | Métodos | Enumeraciones

setInputXMLFromText

```
public void setInputXMLFromText(String inputText)
```

Aporta el contenido del documento XML que se debe validar.
Parámetros: `inputText`: la cadena suministrada es el contenido del documento XML que se debe validar.

Subir | Métodos | Enumeraciones

setInputXMLTextCollection

```
public void setInputXMLTextCollection(Collection<?> stringCollection)
```

Aporta el contenido de varios archivos XML que se usarán como datos de entrada.
Parámetros: `stringCollection`: colección de cadenas de texto, cada una de las cuales es el contenido de un archivo XML.

Subir | Métodos | Enumeraciones

setParallelAssessment

```
public void setParallelAssessment(boolean support)
```

Habilita/deshabilita el uso de la evaluación en paralelo. El valor `true` habilita la evaluación en paralelo; `false` la deshabilita. Valor predeterminado: `false`.
Parámetros:
`support`: toma el valor booleano `true` o `false`.

Subir | Métodos | Enumeraciones

setPythonScriptFile

```
public void setPythonScriptFile(String file)
```

Establece la ubicación en forma de URL del archivo de script Python.
Parámetros: `file`: cadena de texto que debe ser una URL absoluta que dé la ubicación exacta del archivo Python.

Subir | Métodos | Enumeraciones

setSchemaFileCollection

`public void setSchemaFileCollection(Collection<?> fileCollection)`

Aporta la colección de archivos XML que se usarán como esquemas XML externos. Los archivos se identifican por sus URL.

Parámetros: `fileCollection`: colección de cadenas de texto, cada una de las cuales es la URL absoluta de un archivo de esquema XML.

Subir | Métodos | Enumeraciones

setSchemaFileName

`public void setSchemaFileName(String filePath)`

Establece la ubicación en forma de URL del documento de esquema XML que se debe usar.

Parámetros: `filePath`: cadena de texto que debe ser una URL absoluta que dé la ubicación exacta del archivo de esquema XML.

Subir | Métodos | Enumeraciones

setSchemaFromText

`public void setSchemaFromText(String schemaText)`

Aporta el contenido del documento XML que se debe usar.

Parámetros: `schemaText`: cadena de texto que es el contenido del documento de esquema XML que se debe usar.

Subir | Métodos | Enumeraciones

setSchemaImports

`public void setSchemaImports(ENUMSchemaImports opt)`

Indica cómo ocuparse de las importaciones de esquema dependiendo del valor de los atributos de los elementos `xs:import`. La opción se indica mediante el literal de `ENUMSchemaImports` seleccionado.

Parámetros: `opt`: almacena el literal de `ENUMSchemaImports` que determina qué se hace con las importaciones de esquemas. Para más información consulte la descripción de `ENUMSchemaImports`.

Subir | Métodos | Enumeraciones

setSchemalocationHints

`public void setSchemalocationHints(ENUMLoadSchemalocation opt)`

Especifica el mecanismo que se debe usar para encontrar el esquema. El mecanismo viene dado por el literal de `ENUMLoadSchemalocation` seleccionado.

Parámetros: `opt`: almacena el literal de `ENUMLoadSchemalocation` que determina el mecanismo que se debe utilizar. Para más información consulte la descripción de `ENUMLoadSchemalocation`.

 © 2014 Altova GmbH

Subir | Métodos | Enumeraciones

setSchemaMapping

`public void setSchemaMapping(ENUMSchemaMapping opt)`

Establece la asignación que se debe usar para encontrar el esquema. La asignación viene dada por el literal de `ENUMSchemaMapping` seleccionado.

Parámetros: `opt`: almacena el literal de `ENUMSchemaMapping`. Para más información consulte la descripción de `ENUMSchemaMapping`.

Subir | Métodos | Enumeraciones

setInputSchemaTextCollection

`public void setInputSchemaTextCollection(Collection<?> stringCollection)`

Aporta el contenido de varios documentos de esquema XML.

Parámetros: `stringCollection`: colección de cadenas de texto, cada una de las cuales es el contenido de un documento de esquema XML.

Subir | Métodos | Enumeraciones

setStreaming

`public void setStreaming(boolean support)`

Habilita la validación de transmisión por secuencias. En el modo de transmisión por secuencias, los datos almacenados en memoria se minimizan y el procesamiento es más rápido.

Parámetros: `support`: el valor `true` habilita la validación de transmisión por secuencias; `false` la deshabilita. El valor predeterminado es `true`.

Subir | Métodos | Enumeraciones

setXincludeSupport

`public void setXIncludeSupport(boolean support)`

Habilita o deshabilita el uso de elementos `XInclude`. El valor `true` lo habilita; `false` lo deshabilita. El valor predeterminado es `false`.

Parámetros: `support`: el valor binario `true` o `false`.

Subir | Métodos | Enumeraciones

setXMLValidationMode

`public void setXMLValidationMode(ENUMXMLValidationMode mode)`

Establece el modo de validación XML, que es un literal de enumeración de

ENUMXMLValidationMode.
Parámetros: mode: un literal de enumeración de ENUMXMLValidationMode que determina si se valida el documento o si se comprueba su formato XML.

Subir | Métodos | Enumeraciones

setXSDVersion

```
public void setXSDVersion(ENUMXSDVersion version)
```

Establece la versión de XML Schema que se debe usar para validar el documento XML.
Parámetros: version: un literal de enumeración de ENUMXSDVersion que establece la versión de XML Schema.

Subir | Métodos | Enumeraciones

Enumeraciones
ENUMAssessmentMode
ENUMLoadSchemalocation
ENUMSchemaImports
ENUMSchemaMapping
ENUMValidationMode
ENUMValidationType
ENUMWellformedCheckType
ENUMXSDVersion

Subir | Métodos | Enumeraciones

ENUMAssessmentMode

```
public enum ENUMAssessmentMode {
       eAssessmentModeLax
       eAssessmentModeStrict    }
```

ENUMAssessmentMode toma uno de estos literales: eAssessmentModeLax, eAssessmentModeStrict. Definen si la validación debe ser de tipo estricto (strict) o permisivo (lax).

Utilizado por (Interfaz::Método):
XMLValidator setAssessmentMode

Subir | Métodos | Enumeraciones

 © 2014 Altova GmbH

ENUMLoadSchemalocation

```
public enum ENUMLoadSchemalocation {
       eLoadBySchemalocation
       eLoadByNamespace
       eLoadCombiningBoth
       eLoadIgnore    }
```

ENUMLoadSchemalocation contiene el literal de enumeración que especifica el mecanismo para encontrar el esquema. La selección se basa en el atributo schema location del documento de instancia XML o XBRL. Este atributo puede ser xsi:schemaLocation o xsi:noNamespaceSchemaLocation.

- eLoadBySchemalocation utiliza la URL del atributo schema location del archivo XML o XBRL. Este literal de enumeración es el valor predeterminado.
- eLoadByNamespace utiliza la parte de espacio de nombres de xsi:schemaLocation y una cadena vacía en el caso de xsi:noNamespaceSchemaLocation para encontrar el esquema a través de la asignación de catálogos.
- eLoadCombiningBoth: si el URL del espacio de nombres o de schema location tiene una asignación de catálogo, se usa la URL de schema location.
- eLoadCombiningBoth: se pasan por alto tanto el atributo xsi:schemaLocation como el atributo xsi:noNamespaceSchemaLocation.

Utilizado por (Interfaz::Método):

XMLValidator	setSchemalocationHints
XSLT	setSchemalocationHints
XBRL	setSchemalocationHints

Subir | Métodos | Enumeraciones

ENUMSchemaImports

```
public enum ENUMSchemaImports {
       eSILoadBySchemalocation
       eSILoadPreferringSchemalocation
       eSILoadByNamespace
       eSILoadCombiningBoth
       eSILicenseNamespaceOnly    }
```

ENUMSchemaImports contiene el literal de enumeración que define el comportamiento de los elementos xs:import del esquema, cada uno de los cuales tiene un atributo opcional namespace y un atributo opcional schemaLocation.

- eSILoadBySchemalocation usa el valor del atributo schemaLocation para encontrar el esquema, teniendo en cuenta las asignaciones de catálogo. Si está presente el atributo namespace, se importa el espacio de nombres.
- eSILoadPreferringSchemalocation: si está presente el atributo schemaLocation, entonces se usa teniendo en cuenta las asignaciones de catálogo. Si no está presente el atributo schemaLocation, entonces se usa el valor del atributo namespace mediante la asignación de catálogo. Este literal de enumeración es el valor predeterminado.
- eSILoadByNamespace usa el valor del atributo namespace para encontrar el esquema por

medio de una asignación de catálogo.

- eSILoadCombiningBoth: si la URL de namespace o la URL de schemaLocation tiene una asignación de catálogo, ésta se utiliza. Si ambas tienen asignaciones de catálogo, entonces el valor de ENUMSchemaMapping decide qué asignación se utiliza. Si ninguna URL tiene una asignación de catálogo, entonces se usa la URL de schemaLocation.
- eSILicenseNamespaceOnly: se importa el espacio de nombres. No se importa ningún documento de esquema.

Utilizado por (Interfaz::Método):

XMLValidator	setSchemaImports
XSLT	setSchemaImports
XBRL	setSchemaImports

Subir | Métodos | Enumeraciones

ENUMSchemaMapping

```
public enum ENUMSchemaMapping {
       eSMPreferSchemalocation
       eSMPreferNamespace    }
```

ENUMSchemaMapping contiene el literal de enumeración que especifica si se selecciona el espacio de nombres o la ubicación del esquema.

- eSMPreferNamespace: selecciona el espacio de nombres.
- eSMPreferSchemalocation: selecciona la ubicación del esquema. Es el valor predeterminado.

Utilizado por (Interfaz::Método):

XMLValidator	setSchemaMapping
XSLT	setSchemaMapping
XBRL	setSchemaMapping

Subir | Métodos | Enumeraciones

ENUMValidationType

```
public enum ENUMValidationType {
       eValidateAny
       eValidateXMLWithDTD
       eValidateXMLWithXSD
       eValidateDTD
       eValidateXSD }
```

ENUMValidationType contiene el literal de enumeración que especifica si los documentos XML se deben validar con un documento DTD o XSD.

- eValidateAny: el tipo de documento se detecta automáticamente.
- eValidateXMLWithDTD: valida el documento XML con un archivo DTD.
- eValidateXMLWithXSD: valida el documento XML con un archivo XSD.
- eValidateDTD: valida un documento DTD.

 © 2014 Altova GmbH

- eValidateXSD: valida un documento XSD.

Utilizado por (Interfaz::Método):
XMLValidator isValid

Subir | Métodos | Enumeraciones

ENUMWellformedCheckType

```
public enum ENUMWellformedCheckType {
       eWellformedAny
       eWellformedXML
       eWellformedDTD   }
```

ENUMWellformedCheckType contiene el literal de enumeración que especifica de qué tipo es el documento cuyo formato XML debe comprobarse.

- eWellformedAny: el tipo de documento se detecta automáticamente.
- eWellformedXML: comprueba el formato XML de un documento XML.
- eWellformedDTD: comprueba el formato XML de un documento DTD.

Utilizado por (Interfaz::Método):
XMLValidator isWellformed

Subir | Métodos | Enumeraciones

ENUMXSDVersion

```
public enum ENUMXSDVersion {
       eXSDVersionAuto
       eXSDVersion10
       eXSDVersion11   }
```

ENUMXSDVersion contiene el literal de enumeración que especifica la versión de XML Schema.

- eXSDVersionAuto: la versión se detecta automáticamente dependiendo del valor del atributo vc:minVersion del documento XSD. Si el atributo vc:minVersion del documento XSD tiene el valor 1.1, se considera que el documento tiene la versión XSD 1.1. Si este atributo tiene cualquier otro valor o si el atributo no está en el documento, se entiende que el documento tiene la versión XSD 1.0.
- eXSDVersion10: la versión de XML Schema para la validación se establece en XML Schema 1.0.
- eXSDVersion11: la versión de XML Schema para la validación se establece en XML Schema 1.1.

Utilizado por (Interfaz::Método):
XMLValidator setXSDVersion
XSLT setXSDVersion
XQuery setXSDVersion

Subir | Métodos | Enumeraciones

© 2014 Altova GmbH

6.2.3 XSLT

```
public interface XSLT
```

Descripción
Transforma datos XML usando el documento XSLT 1.0, 2.0 o 3.0 dado. Los documentos XML y XSLT se pueden aportar en forma de archivo (con una URL) o de cadena de texto. El resultado se devuelve en forma de archivo (en la ubicación de destino dada) o como cadena de texto. Puede indicar parámetros XSLT y habilitar funciones de extensión de Altova para un procesamiento avanzado (p. ej. generación de gráficos). El documento XSLT también se puede validar. Cuando las cadenas de texto de entrada se deben interpretar como URL, deben utilizarse rutas de acceso absolutas.

A continuación describimos primero los métodos de la interfaz `XSLT` y después sus enumeraciones.

Métodos
Los métodos de la clase se describen por orden alfabético. En la tabla se organizan por grupos para facilitar la consulta.

Procesamiento	XSLT
isValid	setVersion
execute	setXSLFileName
executeAndGetResultAsString	setXSLFromText
executeAndGetResultAsStringWithBaseOutputURI	setaddExternalParameter
getLastErrorMessage	setclearExternalParametersList
setIndentCharacters	setInitialTemplateMode
setStreamingSerialization	setNamedTemplateEntryPoint

XML Schema	XML	Extensiones
setSchemaImports	setInputXMLFileName	setChartExtensionsEnabled
setSchemalocationHints	setInputXMLFromText	setDotNetExtensionsEnabled
setSchemaMapping	setLoadXMLWithPSVI	setJavaExtensionsEnabled
setXSDVersion	setXincludeSupport	setJavaBarcodeExtensionLocation
	setXMLValidationMode	

Subir | **Métodos** | **Enumeraciones**

addExternalParameter

public void **addExternalParameter**(String name, String value)

Añade el nombre y valor de un parámetro externo nuevo. Cada parámetro externo y su valor debe especificarse en una llamada distinta al método. Los parámetros deben declararse en el documento XML. Como los valores de parámetros son expresiones XPath, los valores de parámetros que sean cadenas deben ir entre comillas simples.
Parámetros: name: almacena el nombre del parámetro, que es un QName, como cadena de texto. value: almacena el valor del parámetro como cadena de texto.

Subir | Métodos | Enumeraciones

clearExternalParameterList

public void **clearExternalVariableList()**

Borra la lista de parámetros externos creada con el método AddExternalParameter.

Subir | Métodos | Enumeraciones

execute

public boolean **execute**(String outputFile)

Ejecuta la transformación XSLT de acuerdo con la especificación XSLT indicada en ENUMXSLTVersion (véase el método setVersion) y guarda el resultado en el archivo de salida indicado en el parámetro outputFile. Si ocurre un error, se emite una excepción RaptorXMLException. Use el método getLastErrorMessage para obtener más información.
Parámetros: outputFile: una cadena que indica la ubicación (ruta de acceso y nombre de archivo) del archivo de salida.
Devuelve: el booleano true si la ejecución finaliza correctamente y false si no se puede completar la ejecución.

Subir | Métodos | Enumeraciones

executeAndGetResultAsString

public String **executeAndGetResultAsString()**

Ejecuta la transformación XSLT de acuerdo con la especificación XSLT indicada en ENUMXSLTVersion (véase el método setVersion) y devuelve el resultado en forma de cadena de texto. Si ocurre un error, se emite una excepción RaptorXMLException. Use el método getLastErrorMessage para obtener más información.
Devuelve: una cadena que es el resultado de la transformación XSLT.

Subir | Métodos | Enumeraciones

executeAndGetResultAsStringWithBaseOutputURI

public String **executeAndGetResultAsStringWithBaseOutputURI**(String baseURI)

Ejecuta la transformación XSLT de acuerdo con la especificación XSLT indicada en ENUMXSLTVersion (véase el método setVersion) y devuelve el resultado en forma de cadena de texto en la ubicación definida por el URI base. Si ocurre un error, se emite una excepción RaptorXMLException. Use el método getLastErrorMessage para obtener más información.
Parámetros: baseURI: una cadena que indica el URI.
Devuelve: una cadena que es el resultado de la transformación XSLT.

Subir | Métodos | Enumeraciones

getLastErrorMessage

```
public String getLastErrorMessage()
```

Recupera el último mensaje de error del motor XSLT.
Devuelve: una cadena que es el último mensaje de error del motor XSLT.

Subir | Métodos | Enumeraciones

isValid

```
public boolean isValid()
```

Devuelve el resultado de validar el documento XSLT de acuerdo con la especificación XSLT indicada en ENUMXSLTVersion (véase el método setVersion). El resultado es true si el documento es válido y false si no lo es. Si ocurre un error, se emite una excepción RaptorXMLException. Use el método getLastErrorMessage para obtener más información.
Devuelve: el booleano true si el documento es válido y false si no lo es.

Subir | Métodos | Enumeraciones

setChartExtensionsEnabled

```
public void setChartExtensionsEnabled(boolean enable)
```

Habilita o deshabilita las funciones de extensión de Altova para gráficos.
Parámetros: enable: el valor true habilita las extensiones para gráficos; false las deshabilita. El valor predeterminado es true.

Subir | Métodos | Enumeraciones

setDotNetExtensionsEnabled

```
public void setDotNetExtensionsEnabled(boolean enable)
```

Habilita o deshabilita las funciones de extensión .NET.
Parámetros: enable: el valor true habilita las extensiones .NET; false las deshabilita. El valor predeterminado es true.

Subir | Métodos | Enumeraciones

setJavaBarcodeExtensionLocation

`public void setJavaBarcodeExtensionLocation(String path)`

Especifica la ubicación del archivo de extensión para códigos de barras. Consulte este anexo para obtener más información.

Parámetros: `path`: la cadena indicada debe ser una URL absoluta que dé la ubicación base del archivo que se debe usar.

Subir | Métodos | Enumeraciones

setJavaExtensionsEnabled

`public void setJavaExtensionsEnabled(boolean enable)`

Habilita o deshabilita las funciones de extensión Java.

Parámetros: `enable`: el valor `true` habilita las extensiones Java; `false` las deshabilita. El valor predeterminado es `true`.

Subir | Métodos | Enumeraciones

setIndentCharacters

`public void setIndentCharacters(String chars)`

Establece el carácter que se usará como sangría en el documento de salida.

Parámetros: `chars`: almacena el carácter de sangría.

Subir | Métodos | Enumeraciones

setInitialTemplateMode

`public void setInitialTemplateMode(String mode)`

Establece el nombre del modo de plantilla inicial. El procesamiento comenzará a partir de las plantillas que tengan este valor de modo. La transformación debe iniciarse después de asignar los documentos XML y XSLT.

Parámetros: `mode`: el nombre del modo de plantilla inicial, en forma de cadena de texto.

Subir | Métodos | Enumeraciones

setInputXMLFileName

`public void setInputXMLFileName(String xmlFile)`

Establece la ubicación en forma de URL del documento XML que se debe transformar.

Parámetros: `xmlFile`: la cadena indicada debe ser una URL absoluta que dé la ubicación exacta del archivo XML que se debe utilizar.

 © 2014 Altova GmbH

Subir | Métodos | Enumeraciones

setInputXMLFromText

`public void setInputXMLFromText(String xmlText)`

Aporta el contenido del documento XML de entrada como texto.

Parámetros: `xmlText`: la cadena indicada son los datos XML que se deben procesar.

Subir | Métodos | Enumeraciones

setLoadXMLWithPSVI

`public void setLoadXMLWithPSVI(boolean load)`

Habilita o deshabilita la opción de carga y uso del conjunto Post Schema Validation Infoset (PSVI). Si se carga el PSVI, la información obtenida del esquema se puede usar para calificar datos del documento XML. El valor `true` habilita la carga de PSVI; `false` la deshabilita.

Parámetros: `load`: toma el booleano `true` o `false`.

Subir | Métodos | Enumeraciones

setNamedTemplateEntryPoint

`public void setNamedTemplateEntryPoint(String template)`

Aporta el nombre de la plantilla con nombre con la que se debe iniciar el procesamiento.

Parámetros: `template`: el nombre de la plantilla con nombre, en forma de cadena de texto.

Subir | Métodos | Enumeraciones

setSchemaImports

`public void setSchemaImports(ENUMSchemaImports opt)`

Especifica cómo se deben realizar las importaciones de esquema en función del valor de los atributos de los elementos `xs:import`. Esto se especifica con el literal de `ENUMSchemaImports` seleccionado.

Parámetros: `opt`: almacena el literal de `ENUMSchemaImports` que determina cómo se deben realizar las importaciones de esquema.

Subir | Métodos | Enumeraciones

setSchemalocationHints

`public void setSchemalocationHints(ENUMLoadSchemalocation opt)`

Especifica el mecanismo que se debe usar para encontrar el esquema. El mecanismo viene dado por el literal de `ENUMLoadSchemalocation` seleccionado.

Parámetros: `opt`: almacena el literal de `ENUMLoadSchemalocation` que determina el mecanismo que se debe utilizar para encontrar el esquema.

Subir | Métodos | Enumeraciones

setSchemaMapping

`public void setSchemaMapping(ENUMSchemaMapping opt)`

Establece qué asignación se usa para encontrar el esquema. La asignación viene dada por el literal de `ENUMSchemaMapping` seleccionado.

Parámetros: `opt`: almacena el literal de `ENUMSchemaMapping`.

Subir | Métodos | Enumeraciones

setStreamingSerialization

`public void setStreamingSerialization(boolean support)`

Habilita la serialización de secuencias de datos. En el modo de transmisión por secuencias, los datos almacenados en memoria se minimizan y el procesamiento es más rápido.

Parámetros: `support`: el valor `true` habilita la serialización de secuencias de datos; `false` la deshabilita.

Subir | Métodos | Enumeraciones

setVersion

`public void setVersion(EnumXSLTVersion version)`

Establece la versión XSLT que se debe usar para el procesamiento (validación o transformación XSLT).

Parámetros: `version`: almacena un literal de `EnumXSLTVersion`: `eVersion10`, `eVersion20` o `eVersion30`.

Subir | Métodos | Enumeraciones

setXincludeSupport

`public void setXIncludeSupport(boolean support)`

Habilita o deshabilita el uso de elementos `XInclude`. El valor `true` habilita el uso de elementos XInclude; `false` lo deshabilita. El valor predeterminado es `false`.

Parámetros: `support`: toma el booleano `true` o `false`.

Subir | Métodos | Enumeraciones

setXMLValidationMode

`public void setXMLValidationMode(ENUMXMLValidationMode mode)`

Establece el modo de validación XML, que es un literal de enumeración de

 © 2014 Altova GmbH

ENUMXMLValidationMode.
Parámetros: mode: es un literal de enumeración de ENUMXMLValidationMode.

Subir | Métodos | Enumeraciones

setXSDVersion

```
public void setXSDVersion(ENUMXSDVersion version)
```

Establece la versión de XML Schema con la que se debe validar el documento XML.
Parámetros: version: es un literal de enumeración de ENUMXSDVersion.

Subir | Métodos | Enumeraciones

setXSLFileName

```
public void setXSLFileName(String xslFile)
```

Establece la ubicación en forma de URL del documento XSLT que se debe usar para la transformación.
Parámetros: xslFile: la cadena indicada debe ser una URL absoluta que aporta la ubicación exacta del archivo XSLT.

Subir | Métodos | Enumeraciones

setXSLFromText

```
public void setXSLFromText(String xslText)
```

Aporta el contenido del documento XSLT como texto.
Parámetros: xslText: la cadena indicada es el documento XSLT que se debe usar para la transformación.

Subir | Métodos | Enumeraciones

Enumeraciones
ENUMXSLTVersion

ENUMXSLTVersion

```
public enum ENUMXSLTVersion {
       eVersion10
       eVersion20
       eVersion30 }
```

ENUMXSLTVersion toma uno de estos literales de enumeración: eVersion10, eVersion20,

`eVersion30`. Determinan la versión XSLT que se debe usar para el procesamiento (validación o transformación XSLT).

Utilizado por (Interfaz::Método):

XSLT	setVersion

Subir | **Métodos** | **Enumeraciones**

 © 2014 Altova GmbH

6.2.4 XQuery

```
public interface XQuery
```

Descripción
Ejecuta documentos XQuery 1.0 y 3.0 usando el motor de RaptorXML. Los documentos XQuery y XML pueden darse como archivos (por medio de una URL) o como cadenas de texto. El resultado se devuelve en forma de archivo (en la ubicación de destino indicada) o como cadena de texto. Puede indicar variables XQuery externas y habilitar diferentes opciones de serialización. El documento XQuery también se puede validar. Cuando las cadenas de texto de entrada se deben interpretar como URL, deben utilizarse rutas de acceso absolutas.

A continuación describimos primero los métodos de la interfaz XQuery y después sus enumeraciones.

Métodos
Los métodos de la clase se describen por orden alfabético. En la tabla se organizan por grupos para facilitar la consulta.

Procesamiento	XML	XQuery
isValid	setInputXMLFileName	setVersion
execute	setInputXMLFromText	setXQueryFileName
executeAndGetResultAsString	setLoadXMLWithPSVI	setXQueryFromText
getLastErrorMessage	setXincludeSupport	addExternalVariable
	setXMLValidationMode	clearExternalVariableList
	setXSDVersion	

Opciones de serialización	Extensiones
setIndentCharacters	setChartExtensionsEnabled
setOutputEncoding	setDotNetExtensionsEnabled
setOutputIndent	setJavaExtensionsEnabled
setOutputMethod	
setOutputOmitXMLDeclaration	

Subir | **Métodos** | **Enumeraciones**

addExternalVariable

```
public void addExternalVariable(String name, String value)
```

Añade el nombre y valor de una variable externa nueva. Cada variable externa y su valor se debe especificar en una llama distinta al método. Las variables se deben declarar en el documento XQuery (y la declaración de tipo es opcional). Sea cual sea la declaración de tipo para la variable externa en el documento XQuery, el valor de la variable dado como argumento de `AddExternalVariable` no tiene por qué ser un delimitador (como las comillas, por ejemplo).
Parámetros: `name`: almacena el nombre de la variable, que es un QName, como cadena de texto.
`value`: almacena el valor de la variable como cadena de texto.

Subir | Métodos | Enumeraciones

clearExternalVariableList

```
public void clearExternalVariableList()
```

Borra la lista de variables externas creadas con el método `AddExternalVariable`.

Subir | Métodos | Enumeraciones

execute

```
public boolean execute(String outputFile)
```

Ejecuta la transformación XQuery de acuerdo con la especificación XQuery indicada en `ENUMXQueryVersion` (véase el método `setVersion`) y guarda el resultado en el archivo de salida indicado en el parámetro `outputFile`.
Parámetros: `outputFile`: una cadena que indica la ubicación (ruta de acceso y nombre de archivo) del archivo de salida.
Devuelve: el booleano `true` si la ejecución finaliza correctamente, `false` si no.

Subir | Métodos | Enumeraciones

executeAndGetResultAsString

```
public String executeAndGetResultAsString()
```

Ejecuta la transformación XQuery de acuerdo con la especificación XQuery indicada en `ENUMXQueryVersion` (véase el método `setVersion`) y devuelve el resultado en forma de cadena de texto.
Devuelve: una cadena que es el resultado de la ejecución de XQuery.

Subir | Métodos | Enumeraciones

getLastErrorMessage

```
public String getLastErrorMessage()
```

Recupera el último mensaje de error del motor XQuery.
Devuelve: una cadena que es el último mensaje de error del motor XQuery.

Subir | Métodos | Enumeraciones

isValid

public boolean **isValid()**

Devuelve el resultado de validar el documento XQuery de acuerdo con la especificación XQuery indicada en ENUMXQueryVersion (véase el método setVersion). El resultado es true si el documento es válido y false si no lo es. Si ocurre un error, se emite una excepción RaptorXMLException. Use el método getLastErrorMessage para obtener más información.

Devuelve: el booleano true si el documento es válido, false si no lo es.

Subir | Métodos | Enumeraciones

setChartExtensionsEnabled

public void **setChartExtensionsEnabled(**boolean enable**)**

Habilita o deshabilita las funciones de extensión de Altova para gráficos.

Parámetros: enable: el valor true habilita las extensiones para gráficos; false las deshabilita. El valor predeterminado es true.

Subir | Métodos | Enumeraciones

setDotNetExtensionsEnabled

public void **setDotNetExtensionsEnabled(**boolean enable**)**

Habilita o deshabilita las funciones de extensión .NET.

Parámetros: enable: el valor true habilita las extensiones .NET; false las deshabilita. El valor predeterminado es true.

Subir | Métodos | Enumeraciones

setIndentCharacters

public void **setIndentCharacters(**String chars**)**

Establece el carácter que se usará como sangría en el documento de salida.

Parámetros: chars: almacena el carácter de sangría.

Subir | Métodos | Enumeraciones

setInputXMLFileName

public void **setInputXMLFileName(**String xmlFile**)**

Establece la ubicación en forma de URL del documento XML que se debe usar para la ejecución de XQuery.

Parámetros: xmlFile: la cadena indicada debe ser una URL absoluta que dé la ubicación exacta del archivo XML que se debe usar.

Subir | Métodos | Enumeraciones

setInputXMLFromText

```
public void setInputXMLFromText(String xmlText)
```

Aporta el contenido del documento XML de entrada como texto.
Parámetros: `xmlText`: la cadena indicada son los datos XML que se deben procesar.

Subir | Métodos | Enumeraciones

setJavaExtensionsEnabled

```
public void setJavaExtensionsEnabled(boolean enable)
```

Habilita o deshabilita las funciones de extensión Java.
Parámetros: `enable`: el valor `true` habilita las extensiones Java; `false` las deshabilita. El valor predeterminado es `true`.

Subir | Métodos | Enumeraciones

setLoadXMLWithPSVI

```
public void setLoadXMLWithPSVI(boolean load)
```

Habilita o deshabilita la opción de carga y uso del conjunto Post Schema Validation Infoset (PSVI). Si se carga el PSVI, la información obtenida del esquema se puede usar para calificar datos del documento XML. El valor `true` habilita la carga de PSVI; `false` la deshabilita.
Parámetros: `load`: toma el booleano `true` o `false`.

Subir | Métodos | Enumeraciones

setOutputEncoding

```
public void setOutputEncoding(String encoding)
```

Establece la codificación del documento de salida.
Parámetros: `encoding`: usa un nombre de codificación IANA oficial como `UTF-8`, `UTF-16`, `US-ASCII`, `ISO-8859-1` como cadena de texto.

Subir | Métodos | Enumeraciones

setOutputIndent

```
public void setOutputIndent(boolean indent)
```

Habilita o deshabilita la aplicación de sangría en el documento de salida.
Parámetros: `indent`: el valor `true` habilita la sangría; `false` la deshabilita.

Subir | Métodos | Enumeraciones

 © 2014 Altova GmbH

setOutputMethod

`public void setOutputMethod(String outputMethod)`

Especifica la serialización del documento de salida.

Parámetros: `outputMethod`: los valores válidos son `xml | xhtml | html | text`, dados como cadenas de texto. El valor predeterminado es `xml`.

Subir | Métodos | Enumeraciones

setOutputOmitXMLDeclaration

`public void setOutputOmitXMLDeclaration(boolean omit)`

Habilita o deshabilita la inclusión de la declaración XML en el documento de salida.

Parámetros: `omit`: el valor `true` omite la declaración; `false` la incluye. El valor predeterminado es `false`.

Subir | Métodos | Enumeraciones

setVersion

`public void setVersion(EnumXQueryVersion version)`

Establece la versión de XQuery que se debe usar para el procesamiento (validación o ejecución de XQuery).

Parámetros: `version`: almacena un literal de enumeración de `EnumXQueryVersion`: `eVersion10` o `eVersion30`. El valor predeterminado es `eVersion30ml`.

Subir | Métodos | Enumeraciones

setXincludeSupport

`public void setXIncludeSupport(boolean support)`

Habilita o deshabilita el uso de elementos XInclude. El valor `true` habilita el uso de elementos XInclude; `false` lo deshabilita. El valor predeterminado es `false`.

Parámetros: `support`: toma el booleano `true` o `false`.

Subir | Métodos | Enumeraciones

setXMLValidationMode

`public void setXMLValidationMode(ENUMXMLValidationMode mode)`

Establece el modo de validación XML, que es un literal de enumeración de `ENUMXMLValidationMode`.

Parámetros: `mode`: es un literal de enumeración de `ENUMXMLValidationMode`.

Subir | Métodos | Enumeraciones

setXQueryFileName

```
public void setXQueryFileName(String queryFile)
```

Establece la ubicación en forma de URL del archivo XQuery que debe ejecutarse.
Parámetros: `queryFile`: la cadena indicada debe ser una URL absoluta que dé la ubicación exacta del archivo XML que se debe usar.

Subir | Métodos | Enumeraciones

setXQueryFromText

```
public void setXQueryFromText(String queryText)
```

Aporta el contenido del documento XQuery en forma de texto.
Parámetros: `queryText`: la cadena dada es el documento XQuery que se debe procesar.

Subir | Métodos | Enumeraciones

setXSDVersion

```
public void setXSDVersion(ENUMXSDVersion version)
```

Establece la versión de XML Schema con la que se debe validar el documento XML.
Parámetros: `version`: es un literal de enumeración de `ENUMXSDVersion`.

Subir | Métodos | Enumeraciones

Enumeraciones
ENUMXQueryVersion

ENUMXQueryVersion

```
public enum ENUMXQueryVersion {
       eVersion10
       eVersion30 }
```

`ENUMXQueryVersion` toma uno de estos literales de enumeración: `eVersion10`, `eVersion30`. Determinan la versión de XQuery que se debe usar para el procesamiento (ejecución o validación).

Utilizado por (Interfaz::Método):
XQuery setVersion

Subir | Métodos | Enumeraciones

6.2.5 XBRL

```
public interface XBRL
```

Descripción
Valida el documento de instancia XBRL o de taxonomía XBRL dado. A continuación describimos primero los métodos de la interfaz y después sus enumeraciones.

Clase de utilidad
Se define una clase de utilidad para `FormulaParam`. Almacena dos miembros y un constructor.

ParamValuePair

```
public class ParamValuePair
public class ParamValuePair
    {
        public String paramType;
        public String paramValue;
        public ParamValuePair( String type, String value )
        {
                        paramType = type;
                        paramValue = value;
        }
     };
```

Métodos
Los métodos de la clase se describen por orden alfabético. En la tabla se organizan por grupos para facilitar la consulta.

Según su función

Procesamiento
isValid(ENUM type)
isValid
getLastErrorMessage
setParallelAssessment
setPythonScriptFile

Archivos de entrada
setInputFileName
setInputFileCollection
setInputFromText
setInputTextCollection

Fórmulas y aserciones

 © 2014 Altova GmbH

addAssertionForProcessing
addAssertionSetForProcessing
addFormulaArrayParameter
addFormulaForProcessing
addFormulaParameter(with NS)
addFormulaParameter
addFormulaParameterNamespace
clearFormulaParameterList
readFormulaAssertions
readFormulaOutput
setFormulaAssertionsAsXML
setFormulaAssertionsOutput
setFormulaOutput
evaluateFormula
setFormulaExtensionEnabled
setFormulaPreloadSchemas

Tablas
addTableForProcessing
generateTables
setTableEliminateEmptyRows
setTableExtensionEnabled
setTableOutput
setTableOutputFormat
setTablePreloadSchemas

XML & XML Schema
setXincludeSupport
setSchemaImports
setSchemalocationHints
setSchemaMapping

XBRL en general
setConceptLabelLinkrole
setConceptLabelRole
setGenericLabelLinkrole
setGenericLabelRole
setLabelLang
setDimensionExtensionEnabled
setPreloadSchemas
setTreatXBRLInconsistenciesAsErrors

addAssertionForProcessing

`public void addAssertionForProcessing(String assertion)`

Limita la ejecución de aserciones a la aserción dada. Llame varias veces a este método para especificar más de una aserción.

Parámetros:

`assertion`: la cadena dada almacena el nombre de la aserción. Use `##none` para no procesar ninguna aserción y `##all` para procesarlas todas.

addAssertionSetForProcessing

```
public void addAssertionSetForProcessing(String assertionSet)
```

Limita la ejecución al conjunto de aserciones dado. Llame varias veces a este método para especificar más de un conjunto de aserciones.

Parámetros:

`assertionSet`: la cadena dada almacena el nombre del conjunto de aserciones. Use `##none` para no procesar ningún conjunto de aserciones y `##all` para procesarlos todos.

addFormulaArrayParameter

```
public void addFormulaArrayParameter(String type, String name, Object[]
values)
```

Añade un parámetro matriz utilizado en el proceso de evaluación de fórmulas.

Parámetros:

`type`: una cadena que da el tipo de datos predeterminado de los valores de no pares dentro de los valores de matriz. Tipo predeterminado: `xs:string`.

`name`: una cadena que da el nombre del parámetro.

`values`: una matriz de valores y pares valor/tipo de datos.

Para más información y ejemplos de código consulte el apartado Parámetros de fórmulas XBRL.

addFormulaForProcessing

```
public void addFormulaForProcessing(String formula)
```

Limita la ejecución a la fórmula dada. Llame varias veces a este método para especificar más de una fórmula.

Parámetros:

`formula`: la cadena dada almacena el nombre de la fórmula. Use `##none` para no procesar ninguna fórmula y `##all` para procesarlas todas.

addFormulaParameter (con espacio de nombres) OBSOLETO

```
public void addFormulaParameter(String type, String name, String value, String
 namespace)
```

Añade un parámetro utilizado en el proceso de evaluación de fórmulas.

Parámetros:

`type`: una cadena que indica el tipo de datos del parámetro.

`name`: una cadena que indica el nombre del parámetro.

`value`: una cadena que indica el valor del parámetro.

`namespace`: una cadena que da el espacio de nombres del parámetro.

Nota: si usa este método, el espacio de nombres se pasa a `addFormulaParameterNamespace`.

addFormulaParameter

```
public void addFormulaParameter(String type, String name, String value)
```

Añade un parámetro utilizado en el proceso de evaluación de fórmulas.

Parámetros:

`type`: una cadena que indica el tipo de datos del parámetro.

`name`: una cadena que indica el nombre del parámetro.

`value`: una cadena que indica el valor del parámetro.

 © 2014 Altova GmbH

addFormulaParameterNamespace

`public void addFormulaParameter(String prefix, String URI)`

Define un espacio de nombres utilizado en los QName de nombres, tipos o valores de parámetro.

Parámetros:

`prefix`: el prefijo de espacio de nombres de los valores pasados a `addFormulaArrayParameter`.

`URI`: el URI de espacio de nombres.

addTableForProcessing

`public void addTableForProcessing(String table)`

Limita la generación de tablas a la tabla dada. Llame varias veces a este método para especificar más de una tabla.

Parámetros

`table`: la cadena dad almacena el nombre de la tabla. Use `##none` para no procesar ninguna tabla y `##all` para procesarlas todas.

clearFormulaParameterList

`public void clearFormulaParameterList()`

Borra la lista de parámetros de fórmula creados con el método `addFormulaParameter`.

evaluateFormula

`public boolean evaluateFormula()`

Devuelve el resultado de evaluar las fórmulas XBRL de un archivo de instancia XBRL. El resultado es `true` si la evaluación finaliza correctamente y `false` si no. Si ocurre un error, se emite una excepción `RaptorXMLException`. Use el método `getLastErrorMessage` para obtener más información.

Devuelve:

el booleano `true` si la evaluación finaliza correctamente, `false` si no.

generateTables

`public boolean generateTables()`

Evalúa las tablas XBRL de un archivo de instancia. El resultado es `true` si la evaluación es correcta y `false` si no lo es. Si ocurre un error, se genera `RaptorXMLException`. Use el método `getLastErrorMessage` para obtener más información.

Devuelve:

el booleano `true` si la evaluación es correcta y `false` si no lo es.

getLastErrorMessage

`public String getLastErrorMessage()`

Recupera el último mensaje de error del motor XBRL.

Devuelve: una cadena que es el último mensaje de error del motor XBRL..

isValid

`public boolean isValid(ENUMValidationType type)`

Devuelve el resultado de validar el documento de instancia XBRL o la taxonomía XBRL. El tipo

de documento que se debe validar viene dado por el parámetro `type`, que toma como valor un literal de `ENUMValidationType`. El resultado es `true` si el documento es válido y `false` si no es válido. Si ocurre un error, se emite una excepción `RaptorXMLException`. Use el método `getLastErrorMessage` para obtener más información.
Parámetros:
`type`: un literal de `ENUMValidationType` que indica si se debe validar un documento de instancia XBRL o una taxonomía XBRL.
Devuelve:
el booleano `true` si el documento es válido y `false` si no es válido.

isValid

`public boolean isValid()`

Devuelve el resultado de validar el documento XBRL indicado. El resultado es `true` si el documento es válido y `false` si no es válido.
Devuelve:
el booleano `true` si el documento es válido y `false` si no es válido..

readFormulaAssertions

`public String readFormulaAssertions()`

Recupera las aserciones de fórmula del archivo especificado.
Devuelve: una cadena que contiene las aserciones de fórmula.

readFormulaOutput

`public String readFormulaOutput()`

Evalúa las aserciones de fórmula del archivo especificado y devuelve el resultado de la evaluación.
Devuelve: una cadena que es la evaluación de las aserciones de fórmula.

setConceptLabelLinkrole

`public void setConceptLabelLinkrole(String labelLinkrole)`

Especifica la función de vínculo extendido preferida a la hora de representar etiquetas de concepto.
Parámetros:
`labelLinkrole`: la cadena dada almacena la función de vínculo preferida.

setConceptLabelRole

`public void setConceptLabelRole(String labelRole)`

Especifica la función de etiqueta preferida a la hora de representar etiquetas de concepto.
Parámetros:
`labelRole`: la cadena dada almacena la función de etiqueta preferida. Valor predeterminado: `http://www.xbrl.org/2008/role/label`.

setDimensionExtensionEnabled

`public void setDimensionExtensionEnabled(boolean bEnable)`

Habilita la validación con la extensión XBRL Dimension. El valor `true` habilita esta opción; `false` la deshabilita. El valor predeterminado es `true`.

 © 2014 Altova GmbH

Parámetros:
`bEnable`: toma el booleano `true` o `false`..

setFormulaAssertionsAsXML

`public void setFormulaAssertionsAsXML(boolean bEnable)`

Habilita el formato XML del archivo de aserciones cuando RaptorXML+XBRL se ejecuta con las aserciones habilitadas. El valor `true` habilita el formato de salida XML; `false` genera resultados en formato JSON. El valor predeterminado es `false`.
Parámetros:
`bEnable`: toma el booleano `true` o `false`..

setFormulaAssertionsOutput

`public void setFormulaAssertionsOutput(String outputFile)`

Establece la ubicación del archivo que contiene las aserciones de fórmulas recuperadas.
Parámetros:
`outputFile`: la cadena dada almacena la ruta de acceso completa del archivo de salida.

setFormulaOutput

`public void setFormulaOutput(String outputFile)`

Establece la ubicación del archivo que contiene el resultado de la evaluación de fórmulas.
Parámetros:
`outputFile`: la cadena dada almacena la ruta de acceso completa del archivo de salida.

setFormulaExtensionEnabled

`public void setFormulaExtensionEnabled(boolean bEnable)`

Habilita la validación con la extensión XBRL Formula. El valor `true` habilita esta opción; `false` la deshabilita. El valor predeterminado es `true`.
Parámetros:
`bEnable`: toma el booleano `true` o `false`.

setFormulaPreloadSchemas

`public void setFormulaPreloadSchemas(boolean bEnable)`

Determina si los esquemas de XBRL Formula se cargan previamente o no. El valor `true` defina la carga previa de estos esquemas; `false` no carga los esquemas. El valor predeterminado es `false`.
Parámetros:
`bEnable`: toma el booleano `true` o `false`.

setGenericLabelLinkrole

`public void setGenericLabelLinkrole(String labelLinkrole)`

Especifica la función de vínculo extendido preferida a la hora de presentar etiquetas genéricas.
Parámetros:
`labelLinkrole`: la cadena dada almacena la función de vínculo preferida.

setGenericLabelRole

```
public void setGenericLabelRole(String labelRole)
```

Especifica la función de etiqueta preferida a la hora de representar etiquetas genéricas.

Parámetros:

`labelRole`: la cadena dada almacena la función de etiqueta preferida. Valor predeterminado: `http://www.xbrl.org/2008/role/label`.

setInputFileCollection

```
public void setInputFileCollection(Collection<?> fileCollection)
```

Suministra la colección de archivos XBRL que se usarán como datos de entrada. Los archivos se identifican con sus URL.

Parámetros:

`fileCollection`: una colección de cadenas de texto, cada una de las cuales es la URL absoluta de un archivo XBRL de entrada.

setInputFileName

```
public void setInputXMLFileName(String filePath)
```

Establece la ubicación en forma de URL del documento XBRL que se debe validar.

Parámetros:

`filePath`: la cadena indicada debe ser una URL absoluta que indica la ubicación exacta del archivo XBRL.

setInputFromText

```
public void setInputFromText(String inputText)
```

Suministra el contenido del documento XBRL en forma de texto.

Parámetros:

`inputText`: la cadena indicada es el contenido del documento XBRL que se debe validar.

setInputTextCollection

```
public void setInputTextCollection(Collection<?> stringCollection)
```

Suministra el contenido de varios archivos XBRL que se usará como datos de entrada.

Parámetros:

`stringCollection`: una colección de cadenas de texto, cada una de las cuales es el contenido de un documento XBRL de entrada.

setLabelLang

```
public void setLabelLang(String labelLang)
```

Especifica el idioma de etiqueta preferido a la hora de representar etiquetas.

Parámetros:

`labelLang`: la cadena dada almacena el idioma de etiqueta preferido. Predeterminado: `en`.

setParallelAssessment

```
public void setParallelAssessment(boolean support)
```

Habilita o deshabilita el uso de la evaluación en paralelo. El valor `true` habilita la evaluación en paralelo; `false` la deshabilita. El valor predeterminado: `false`.

Parámetros:

`support`: toma el valor booleano `true` o `false`.

 © 2014 Altova GmbH

setPreloadSchemas

`public void setPreloadSchemas(boolean preload)`

Determina si los esquemas de XBRL 2.1 se cargan previamente o no. El valor `true` defina la carga previa de estos esquemas; `false` no carga los esquemas. El valor predeterminado es `true`.

Parámetros:

`support`: toma el booleano `true` o `false`. El valor predeterminado es `false`.

setPythonScriptFile

`public void setPythonScriptFile(String file)`

Establece la ubicación en forma de URL del archivo de script Python.

Parámetros:

`file`: la cadena indicada debe ser una URL absoluta que dé la ubicación exacta del archivo Python.

setSchemaImports

`public void setSchemaImports(ENUMSchemaImports opt)`

Indica cómo ocuparse de las importaciones de esquema dependiendo del valor de los atributos de los elementos `xs:import`. La opción se indica mediante el literal de `ENUMSchemaImports` seleccionado.

Parámetros:

`opt`: almacena el literal de `ENUMSchemaImports` que determina qué se hace con las importaciones de esquemas. Para más información consulte la descripción de `ENUMSchemaImports`.

setSchemalocationHints

`public void setSchemalocationHints(ENUMLoadSchemalocation opt)`

Especifica el mecanismo que se debe usar para encontrar el esquema. El mecanismo viene dado por el literal de `ENUMLoadSchemalocation` seleccionado.

Parámetros:

`opt`: almacena el literal de `ENUMLoadSchemalocation` que determina el mecanismo que se debe utilizar para encontrar el esquema. Para más información consulte la descripción de `ENUMLoadSchemalocation`.

setSchemaMapping

`public void setSchemaMapping(ENUMSchemaMapping opt)`

Establece qué asignación se usa para encontrar el esquema. La asignación viene dada por el literal de `ENUMSchemaMapping` seleccionado.

Parámetros:

`opt`: almacena el literal de `ENUMSchemaMapping`. Para más información consulte la descripción de `ENUMSchemaMapping`.

setTableEliminateEmptyRows

`public void setTableEliminateEmptyRows(boolean bEnable)`

Habilita la eliminación de columnas/filas vacías de tablas en el formato de salida HTML. El valor `true` la habilita; `false` la deshabilita.

Parámetros:
`bEnable`: toma el valor booleano `true` o `false`.

setTableExtensionEnabled

`public void setTableExtensionEnabled(boolean bEnable)`

Habilita las extensiones XBRL Table 1.0 para la validación. El valor `true` las habilita; `false` las deshabilita.
Parámetros:
`bEnable`: toma el valor booleano `true` o `false`.

setTableOutput

`public void setTableOutput(String outputFile)`

Establece la ubicación del archivo que contiene el resultado de la generación de tablas.
Parámetros:
`outputFile`: la cadena dada almacena la ruta de acceso completa del archivo de salida.

setTableOutputFormat

`public void setTableOutputAsXML(ENUMTableOutputFormat format)`

Define el formato del archivo de tablas de salida. El formato será el valor de `ENUMTableOutputFormat`.
Parámetros:
`format`: almacena el valor de `ENUMTableOutputFormat`.

setTablePreloadSchemas

`public void setTablePreloadSchemas(boolean bEnable)`

Habilita la carga previa de esquemas de la especificación XBRL Table 1.0. El valor `true` lo habilita; `false` lo deshabilita. Valor predeterminado: `false`.
Parámetros:
`bEnable`: toma el booleano `true` o `false`.

setTreatXBRLInconsistenciesAsErrors

`public void setTreatXBRLInconsistenciesAsErrors(boolean treat)`

El valor `true` hace que la validación XBRL no finalice correctamente si el archivo contiene incoherencias definidas por la especificación XBRL 2.1. El valor predeterminado es `false`. Si el valor es `false`, las incoherencias XBRL definidas por la especificación XBRL 2.1 no se consideran errores.
Parámetros:
`support`: toma el booleano `true` o `false`.

setXincludeSupport

`public void setXIncludeSupport(boolean support)`

Habilita o deshabilita el uso de elementos `XInclude`. El valor `true` habilita el uso de elementos XInclude; `false` lo deshabilita. El valor predeterminado es `false`.
Parámetros:
`support`: toma el booleano `true` o `false`.

 © 2014 Altova GmbH

Subir | Métodos | Enumeraciones

Enumeraciones

ENUMValidationType

```
public enum ENUMValidationType {
        eValidateAny
        eValidateInstance
        eValidateTaxonomy }
```

ENUMValidationType contiene el literal de enumeración que especifica qué tipo de validación debe llevarse a cabo y, en caso de tratarse de un documento XML, si la validación se realiza con un documento DTD o XSD.

- eValidateAny: la versión se detecta automáticamente.
- eValidateInstance: valida un documento de instancia XBRL (.xbrl).
- eValidateTaxonomy: valida una taxonomía XBRL (.xsd).

Utilizado por (Interfaz::Método):

XBRL	isValid

ENUMTableOutputFormat

```
public enum ENUMTableOutputFormat {
        eFormatXML
        eFormatHTML }
```

ENUMTableOutputFormat contiene el literal de enumeración que especifica el formato de salida del documento que contiene las tablas generadas.

- eFormatXML: el documento de salida con las tablas generadas está en formato XML.
- eFormatHTML: el documento de salida con las tablas generadas está en formato HTML.

Utilizado por (Interfaz::Método):

XBRL	setTableOutputFormat

Subir | Métodos | Enumeraciones

6.2.6 RaptorXMLException

```
public interface RaptorXMLException
```

Descripción
Tiene un solo método que genera la excepción.

RaptorXMLException

```
public void RaptorXMLException(String message)
```

Genera una excepción con información sobre el error que se produce durante el procesamiento.

Parámetros: `message`: una cadena con información sobre el error.

 © 2014 Altova GmbH

Altova RaptorXML+XBRL Server 2014

Interfaces COM y .NET

7 Interfaces COM y .NET

Dos interfaces, una sola API
Las interfaces COM y .NET de RaptorXML+XBRL Server utilizan una sola API: la API de COM/ .NET API para RaptorXML+XBRL Server. La interfaz .NET está construida como un contenedor alrededor de la interfaz COM.

Puede usar RaptorXML con:

- Lenguajes de scripting como JavaScript, a través de la interfaz COM
- Lenguajes de programación como C#, a través de la interfaz .NET

Apartados de esta sección
Esta sección se divide en varios apartados:

- Notas sobre la interfaz COM: funcionamiento general de la interfaz COM y pasos necesarios para trabajar con ella.
- Notas sobre la interfaz .NET: cómo preparar el entorno de desarrollo para trabajar con la interfaz .NET.
- Lenguajes de programación: fragmentos de código en los lenguajes de programación más frecuentes, que muestran cómo se llama a las funciones de RaptorXML.
- Referencia de API: documentación del modelo de objetos, de los objetos y de las propiedades de la API.

 © 2014 Altova GmbH

7.1 Notas sobre la interfaz COM

Durante la instalación, RaptorXML+XBRL Server se registra automáticamente como objeto de servidor COM. Esto permite invocar a RaptorXML+XBRL Server desde otras aplicaciones y lenguajes de scripting compatibles con el uso de llamadas COM. Si quiere cambiar la ubicación del paquete de instalación de RaptorXML+XBRL Server, es mejor desinstalar RaptorXML+XBRL Server y volver a instalarlo en su nueva ubicación. De este modo el programa de instalación se encarga del proceso de registro.

Comprobar si RaptorXML+XBRL Server se registró correctamente
Si RaptorXML+XBRL Server se registró correctamente, el registro incluye las clases `RaptorXML.Server`. Estas clases suelen estar en `HKEY_LOCAL_MACHINE\SOFTWARE\Classes`.

Código de ejemplo
El ejemplo de código VBScript muestra cómo se puede usar la API de RaptorXML a través de su interfaz COM. Para este fragmento de código también ofrecemos un archivo de ejemplo en la carpeta `examples/API` de la carpeta de aplicación de RaptorXML.

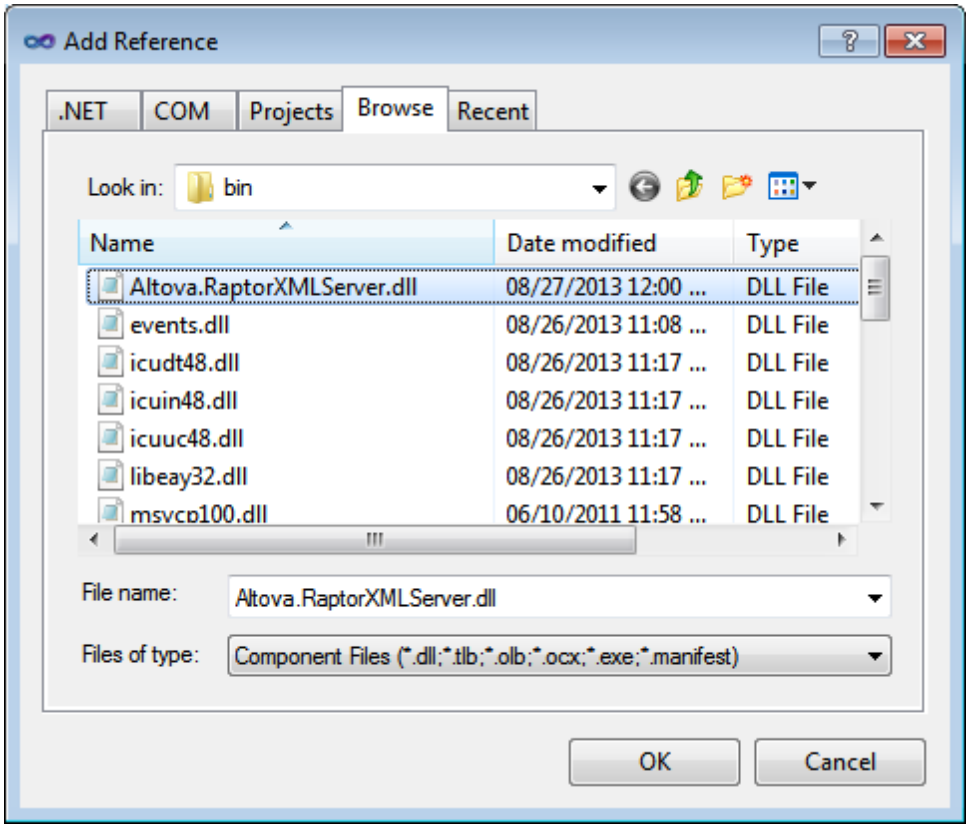
7.2 Notas sobre la interfaz NET

La interfaz .NET está construida como contenedor alrededor de la interfaz COM de RaptorXML. Se ofrece como ensamblado de interoperabilidad principal firmado por Altova y usa el espacio de nombres `Altova.RaptorXMLServer`.

Añadir al proyecto Visual Studio .NET una referencia al DLL de RaptorXML
Para poder usar RaptorXML en su proyecto .NET debe añadir una referencia al DLL de RaptorXML (`Altova.RaptorXMLServer.dll`) en el proyecto. El paquete de instalación de RaptorXML+XBRL Server contiene un archivo DLL firmado y llamado `Altova.RaptorXMLServer.dll`, que se añadirá automáticamente al caché global de ensamblados cuando se instale RaptorXML usando el programa de instalación de RaptorXML. (Su ubicación suele ser la carpeta `C:\WINDOWS\assembly`.)

Para añadir una referencia a este DLL en un proyecto .NET:

1. Abra el proyecto .NET y haga clic en **Proyecto | Añadir referencia**. Aparece el cuadro de diálogo "Añadir referencia" (*imagen siguiente*).



2. En la pestaña *Browse* vaya a la carpeta: `<RaptorXML application folder>/bin`, seleccione el DLL de RaptorXML `Altova.RaptorXMLServer.dll` y haga clic en **Aceptar**.
3. Seleccione el comando **View | Object Browser** para ver los objetos de la API de RaptorXML.

 © 2014 Altova GmbH

Cuando `Altova.RaptorXMLServer.dll` esté a disposición de la interfaz .NET y RaptorXML esté registrado como objeto servidor COM, las funciones de RaptorXML estarán disponibles en el proyecto .NET.

Nota: RaptorXML se registra automáticamente como objeto servidor COM durante la instalación. No es necesario registrarlo a mano.

Nota: si recibe un error de acceso, compruebe que tiene los permisos necesarios. Vaya a Servicios de componentes y otorge permisos a la misma cuenta que ejecuta el grupo de aplicaciones que contiene RaptorXML.

Código de ejemplo

El ejemplo de código C# y de código Visual Basic .NET muestran cómo se puede usar la API de RaptorXML a través de su interfaz .NET. Para estos ejemplos de código también ofrecemos archivos de muestra en la carpeta `examples/API` de la carpeta de aplicación de RaptorXML.

7.3 Lenguajes de programación

No todos los lenguajes de programación ofrecen la misma compatibilidad con COM y .NET. A continuación puede ver algunos ejemplos para los lenguajes de programación más frecuentes. Los fragmentos de código que aparecen en este apartado explican cómo acceder a funciones básicas. Estas funciones básicas están incluidas en los archivos de la carpeta `examples/API` de la carpeta de aplicación RaptorXML+XBRL Server.

VBScript
VBScript se puede usar para acceder a la API de COM de RaptorXML+XBRL Server. El ejemplo de código VBScript demuestra estas funciones básicas:

- Conectarse a la API de COM de RaptorXML+XBRL Server
- Validar un archivo XML
- Realizar una transformación XSLT
- Ejecutar un XQuery

C#
C# se puede usar para acceder a la API de .NET de RaptorXML+XBRL Server. El ejemplo de código C# demuestra cómo acceder a la API para utilizar estas funciones básicas:

- Conectarse a la API de .NET de RaptorXML+XBRL Server
- Validar un archivo XML
- Realizar una transformación XSLT
- Ejecutar un XQuery

Visual Basic .NET
Visual Basic.NET solo difiere de C# en la sintaxis. El acceso a la API de .NET funciona exactamente igual. El ejemplo de código Visual Basic demuestra estas funciones básicas:

- Conectarse a la API de .NET de RaptorXML+XBRL Server
- Validar un archivo XML
- Realizar una transformación XSLT
- Ejecutar un XQuery

Este apartado contiene estos ejemplos:

Para la interfaz COM

- Un ejemplo de código VBScript

Para la interfaz .NET

- Un ejemplo de código C#
- Un ejemplo de código Visual Basic

 © 2014 Altova GmbH

7.3.1 Ejemplo de COM: VBScript

Este ejemplo de código VBScript se divide en varias partes:

- Preparación e inicialización del objeto COM de RaptorXML
- Validación de un archivo XML
- Transformación XSLT y devolución del resultado en forma de cadena de texto
- Procesamiento de un documento XQuery y devolución del resultado en forma de archivo
- Preparación de la secuencia de ejecución del script y de su punto de entrada

---

```
' El objeto COM de RaptorXML
dim objRaptor

' Inicializar el objeto COM de RaptorXML
sub Init
	objRaptor = Null
	On Error Resume Next
	' Intentar cargar el objeto COM de 32 bits. No emitir excepciones si no
se encuentra el objeto
	Set objRaptor = WScript.GetObject( "", "RaptorXML.Server" )
	On Error Goto 0
	if ( IsNull( objRaptor ) ) then
		' Intentar cargar el objeto COM de 64 bits (se emite una excepción
si no se encuentra el objeto)
		Set objRaptor = WScript.GetObject( "", "RaptorXML_x64.Server" )
	end if
	' Configurar el servidor: notificación de errores, nombre y puerto del
servidor HTTP (IPv6 localhost en este	ejemplo)
	objRaptor.ErrorLimit = 1
	objRaptor.ReportOptionalWarnings = true
	objRaptor.ServerName = "::1"
	objRaptor.ServerPort = 8087
end sub
```

```
' Validar un archivo
sub ValidateXML
	' Obtener una instancia del validador del objeto servidor
	dim objXMLValidator
	Set objXMLValidator = objRaptor.GetXMLValidator()

	' Configurar los datos de entrada
	objXMLValidator.InputXMLFileName = "MiArchivoXML.xml"

	' Validar; en caso de que el archivo no sea válido, notificar el problema
devuelto por RaptorXML
	if ( objXMLValidator.IsValid() ) then
		MsgBox( "La cadena de entrada es válida" )
	else
		MsgBox( objXMLValidator.LastErrorMessage )
```

 © 2014 Altova GmbH

```
        end if
end sub

---

' Realizar una transformación; devolver el resultado en forma de cadena de texto
sub RunXSLT
        ' Obtener una instancia del motor XSLT del objeto servidor
        dim objXSLT
        set objXSLT = objRaptor.GetXSLT

        ' Configurar los datos de entrada
        objXSLT.InputXMLFileName = "MiArchivoXML.xml"
        objXSLT.XSLFileName = "MiTransformación.xsl"

        ' Ejecutar la transformación; si finaliza correctamente, el resultado se
devuelve. Si no, el motor devuelve  una lista de errores
        MsgBox( objXSLT.ExecuteAndGetResultAsString() )
end sub

---

' Ejecutar un XQuery; guardar el resultado en un archivo
sub RunXQuery
        ' Obtener una instancia del motor XQuery del objeto servidor
        dim objXQ
        set objXQ = objRaptor.GetXQuery()

        ' Configurar los datos de entrada
        objXQ.InputXMLFileName = "MiArchivoXML.xml"
        objXQ.XQueryFileName = "MiXQuery.xq"

        ' Configurar la serialización (opcional si quiere perfeccionar el formato
del archivo de salida)
        objXQ.OutputEncoding = "UTF8"
        objXQ.OutputIndent = true
        objXQ.OutputMethod = "xml"
        objXQ.OutputOmitXMLDeclaration = false

        ' Ejecutar la consulta; el resultado se serializa en la ruta de acceso
indicada
        call objXQ.Execute( "MyQueryResult.xml" )
end sub

---

' Realizar todas las funciones de muestra
sub main
        Init
        ValidateXML
        RunXSLT
        RunXQuery
end sub

' Punto de entrada del script; ejecutar la función principal
```

```
main
```


© 2014 Altova GmbH

7.3.2 Ejemplo de .NET: C#

Este ejemplo de código C# se divide en varias partes:

- Preparación e inicialización del objeto .NET de RaptorXML
- Validación de un archivo XML
- Transformación XSLT y devolución del resultado en forma de cadena de texto
- Procesamiento de un documento XQuery y devolución del resultado en forma de archivo
- Preparar la secuencia de ejecución del código y su punto de entrada

---

```
using System;
using System.Text;
using Altova.RaptorXMLServer;

namespace RaptorXMLRunner
{
    class Program
    {
    // El objeto .NET de RaptorXML
       static ServerClass objRaptorXMLServer;

    // Inicializar el objeto .NET de RaptorXML
       static void Init()
       {
       // Asignar un objeto de RaptorXML
          objRaptorXMLServer = new ServerClass();

       // Configurar el servidor: notificación de errores, nombre y puerto del
servidor HTTP
       // (IPv6 localhost en este ejemplo)
          objRaptorXMLServer.ErrorLimit = 1;
          objRaptorXMLServer.ReportOptionalWarnings = true;
          objRaptorXMLServer.ServerName = "::1"
          objRaptorXMLServer.ServerPort = 8087
       }
```

```
       // Validar un archivo
          static void ValidateXML()
          {
          // Obtener una instancia del motor de validación del objeto servidor
             XMLValidator objXMLValidator =
objRaptorXMLServer.GetXMLValidator();

          // Configurar los datos de entrada
             objXMLValidator.InputXMLFileName = "MyXMLFile.xml";

          // Validar; si el archivo no es válido,
```

```
            notificar el problema devuelto por RaptorXML
            if ( objXMLValidator.IsValid() )
               Console.WriteLine( "La cadena de entrada no es válida" );
            else
               Console.WriteLine( objXMLValidator.LastErrorMessage );
         }
```

```
      // Realizar una transformación XSLT y
      // devolver el resultado en forma de cadena de texto
         static void RunXSLT()
         {
         // Obtener una instancia del motor XSLT del objeto servidor
            XSLT objXSLT = objRaptorXMLServer.GetXSLT();

            // Configurar los datos de entrada
               objXSLT.InputXMLFileName = "MiArchivoXML.xml";
               objXSLT.XSLFileName = "MiTransformación.xsl";

           // Ejecutar la transformación.
           // Si se ejecuta correctamente, el resultado se devuelve.
           // Si hay errores, se emite una lista de errores
              Console.WriteLine( objXSLT.ExecuteAndGetResultAsString() );
         }
```

```
      // Ejecutar un XQuery y guardar el resultado en un archivo
         static void RunXQuery()
         {
         // Obtener una instancia del motor XQuery del objeto servidor
            XQuery objXQuery = objRaptorXMLServer.GetXQuery();

           // Configurar los datos de entrada
              objXQuery.InputXMLFileName = exampleFolder + "simple.xml";
              objXQuery.XQueryFileName = exampleFolder + "CopyInput.xq";

          // Configurar la serialización (opcional si quiere mejorar el formato
de salida)
              objXQuery.OutputEncoding = "UTF8"
              objXQuery.OutputIndent = true
              objXQuery.OutputMethod = "xml"
              objXQuery.OutputOmitXMLDeclaration = false

          // Ejecutar la consulta. El resultado se serializa en la ruta de
acceso indicada
              objXQuery.Execute( "MyQueryResult.xml" );
         }
```

```
            static void Main(string[] args)
            {
              try
              {
              // Punto de entrada. Realizar todas las funciones
                 Init();
                 ValidateXML();
                 RunXSLT();
                 RunXQuery();
              }
              catch (System.Exception ex)
              {
                  Console.WriteLine( ex.Message );
                  Console.WriteLine( ex.ToString() );
              }
          }
      }
  }
```

7.3.3 Ejemplo de .NET: Visual Basic .NET

Este ejemplo de código Visual Basic se divide en varias partes:

- Preparar e inicializar el objeto .NET de RaptorXML
- Validar un archivo XML
- Transformación XSLT y devolución del resultado en forma de cadena de texto
- Procesamiento de un documento XQuery y devolución del resultado en forma de archivo
- Preparación de la secuencia de ejecución del código y de su punto de entrada

---

```
Option Explicit On
Imports Altova.RaptorXMLServer

Module RaptorXMLRunner

   ' El objeto .NET de RaptorXML
     Dim objRaptor As Server

   ' Inicializar el objeto .NET de RaptorXML
     Sub Init()

      ' Asignar un objeto RaptorXML
        objRaptor = New Server()

      ' Configurar el servidor: notificación de errores, nombre y puerto del
servidor HTTP (IPv6 localhost en este ejemplo)
        objRaptor.ErrorLimit = 1
        objRaptor.ReportOptionalWarnings = True
        objRaptor.ServerName = "::1"
        objRaptor.ServerPort = 8087
     End Sub
```

```
   ' Validar un archivo
     Sub ValidateXML()

      ' Obtener una instancia del validador del objeto RaptorXML
        Dim objXMLValidator As XMLValidator
        objXMLValidator = objRaptor.GetXMLValidator()

      ' Configurar los datos de entrada
        objXMLValidator.InputXMLFileName = "MiArchivoXML.xml"

      ' Validar; si el archivo no es válido, notificar el problema devuelto por
RaptorXML
        If (objXMLValidator.IsValid()) Then
           Console.WriteLine("La cadena de entrada es válida")
        Else
           Console.WriteLine(objXMLValidator.LastErrorMessage)
        End If
     End Sub
```

 © 2014 Altova GmbH

```
   ' Realizar una transformación; devolver el resultado en forma de cadena de
texto
     Sub RunXSLT()

      ' Obtener una instancia del motor XSLT del objeto servidor
        Dim objXSLT As XSLT
        objXSLT = objRaptor.GetXSLT()

      ' Configurar los datos de entrada
        objXSLT.InputXMLFileName = "MiArchivoXML.xml"
        objXSLT.XSLFileName = "MiTransformación.xsl"

      ' Ejecutar la transformación; si se ejecuta correctamente, se devuelve el
resultado; si hay errores, devolver  una lista de errores
        Console.WriteLine(objXSLT.ExecuteAndGetResultAsString())
     End Sub
```

```
   ' Ejecutar un XQuery; guardar el resultado en un archivo
     Sub RunXQuery()

      ' Obtener una instancia del motor XQuery del objeto servidor
        Dim objXQ As XQuery
        objXQ = objRaptor.GetXQuery()

      ' Configurar los datos de entrada
        objXQ.InputXMLFileName = "MyXMLFile.xml"
        objXQ.XQueryFileName = "MyQuery.xq"

      ' Configurar la serialización (opcional para mejorar el formato del
archivo de salida)
        objXQ.OutputEncoding = "UTF8"
        objXQ.OutputIndent = true
        objXQ.OutputMethod = "xml"
        objXQ.OutputOmitXMLDeclaration = false

      ' Ejecutar la consulta; el resultado se serializa en la ruta de acceso
indicada
        objXQ.Execute( "MyQueryResult.xml" )
     End Sub
```

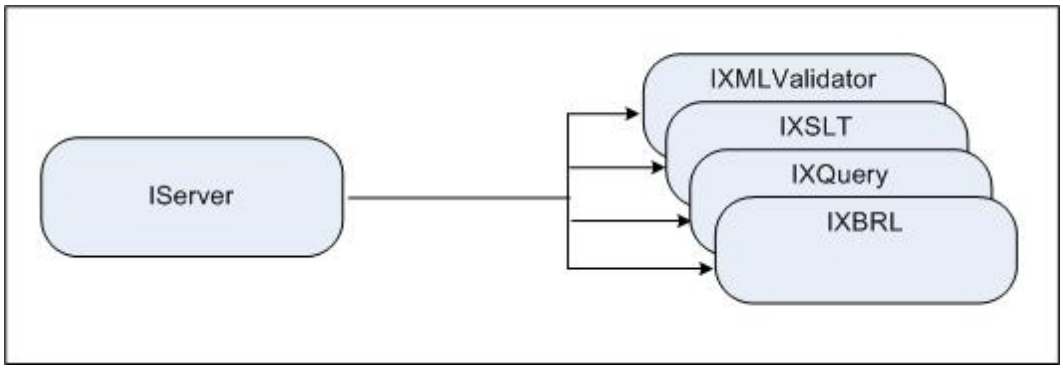
```
     Sub Main()
      ' Punto de entrada; realizar todas las funciones de muestra
        Init()
        ValidateXML()
        RunXSLT()
        RunXQuery()
     End Sub

End Module
```

7.4 Referencia de la API

Esta sección describe la especificación de la API: su modelo de objetos y los detalles de sus interfaces y enumeraciones.

El punto de partida para usar las funciones de RaptorXML es la interfaz `IServer`. Este objeto contiene los objetos que ofrecen las funciones de RaptorXML: validación XML, validación XBRL, transformaciones XSLT y procesamiento de documentos XQuery. El modelo de objetos de la API de RaptorXML aparece en este esquema:



La jerarquía del modelo de objetos aparece más abajo y las interfaces se describen una por una en los apartados siguientes. Los métodos y propiedades de cada interfaz también se describen en detalle.

```
-- IServer
  |-- IXMLValidator
  |-- IXSLT
  |-- IXQuery
  |-- IXBRL
```

 © 2014 Altova GmbH

7.4.1 Interfaces

Las interfaces definidas son estas:

IServer

IXMLValidator

IXSLT

IXQuery

IXBRL

IServer

La interfaz **IServer** ofrece métodos para devolver las interfaces del motor de RaptorXML correspondiente: del validador XML, XBRL, XSLT y XQuery. Las propiedades definen los parámetros de la interfaz.

Métodos
IXMLValidator
IXBRL
IXSLT
IXQuery

Propiedades		
APIMajorVersion	GlobalResourcesFile	ServerName
APIMinorVersion	Is64Bit	ServerPath
APIServicePackVersion	MajorVersion	ServerPort
ErrorFormat	MinorVersion	ServicePackVersion
ErrorLimit	ProductName	UserCatalog
GlobalCatalog	ProductNameAndVersion	
GlobalResourceConfig	ReportOptionalWarnings	

Métodos

Los métodos de la **IServer** devuelven interfaces del correspondiente motor de RaptorXML: del validador XML, XBRL, XSLT y XQuery.

IXMLValidator **GetXMLValidator()** [Subir | Métodos | Propiedades]

Devuelve una instancia del motor del validador XML.

IXBRL **GetXBRL()** [Subir | Métodos | Propiedades]
Devuelve una instancia del motor XBRL.

IXSLT **GetXSLT()** [Subir | Métodos | Propiedades]
Devuelve una instancia del motor XSLT.

IXQuery **GetXQuery()** [Subir | Métodos | Propiedades]
Devuelve una instancia del motor XQuery.

Propiedades
Las propiedades de la interfaz **IServer** se describen a continuación por orden alfabético. La tabla organiza las propiedades por grupos para facilitar su consulta. Recuerde que las cadenas de entrada que deben interpretarse como direcciones URL deben dar rutas de acceso absolutas. Si se usa una ruta de acceso relativa, debe definir en el módulo de llamada un mecanismo para resolver la ruta de acceso relativa.

Errores y advertencias	Catálogos	Recursos globales	Servidor HTTP
ErrorFormat	GlobalCatalog	GlobalResourceConfig	ServerName
ErrorLimit	UserCatalog	GlobalResourcesFile	ServerPath
ReportOptionalWarnings			ServerPort

Información sobre el producto	
ProductName	Is64Bit
ProductNameAndVersion	APIMajorVersion
MajorVersion	APIMinorVersion
MinorVersion	APIServicePackVersion
ServicePackVersion	

int **APIMajorVersion** [Subir | Métodos | Propiedades]
Devuelve la versión principal de la API como número entero. La versión principal de la API puede ser distinta a la versión principal del producto si la API está conectada a otro servidor.

---

int **APIMinorVersion** [Subir | Métodos | Propiedades]

 © 2014 Altova GmbH

Devuelve la versión secundaria de la API como número entero. La versión principal de la API puede ser distinta a la versión secundaria del producto si la API está conectada a otro servidor.

`int` **APIServicePackVersion** [Subir | Métodos | Propiedades]
Devuelve la versión del service pack de la API como número entero. La versión del service pack de la API puede ser distinta a la versión del service pack del producto si la API está conectada a otro servidor.

ENUMErrorFormat **ErrorFormat** [Subir | Métodos | Propiedades]
Establece el formato de los errores de RaptorXML y es un literal de ENUMErrorFormat (`Text | ShortXML | LongXML`).

`int` **ErrorLimit** [Subir | Métodos | Propiedades]
Configura el límite de errores de validación de RaptorXML. Es de tipo `uint`. Si se alcanza el límite de errores, la ejecución se detiene. El valor predeterminado es `100`.

`string` **GlobalCatalog** [Subir | Métodos | Propiedades]
Indica la ubicación del archivo de catálogo principal (punto de entrada). La cadena indicada debe ser una URL absoluta que dé la ubicación exacta del archivo de catálogo que se debe usar.

`string` **GlobalResourceConfig** [Subir | Métodos | Propiedades]
Indica la configuración activa del recurso global que se debe utilizar.

`string` **GlobalResourcesFile** [Subir | Métodos | Propiedades]
Indica el archivo de recurso global. la cadena indicada debe ser una URL absoluta que dé la ubicación exacta del archivo de recursos globales que se debe usar.

`bool` **Is64Bit** [Subir | Métodos | Propiedades]
Comprueba si la aplicación es un ejecutable de 64 bits. *Ejemplo:* para `Altova RaptorXML+XBRL Server 2014r2sp1(x64)`, devuelve `true`.

`int` **MajorVersion** [Subir | Métodos | Propiedades]
Devuelve la versión principal del producto como número entero. *Ejemplo:* para `Altova RaptorXML +XBRL Server 2014r2sp1(x64)`, devuelve `16` (la diferencia entre la versión principal (`2014`) y el año inicial `1998`).

`int` **MinorVersion** [Subir | Métodos | Propiedades]

Devuelve la versión secundaria del producto como número entero. *Ejemplo:* para `Altova RaptorXML+XBRL Server 2014r2sp1(x64)`, devuelve `2` (del número de la versión secundaria `r2`).

`string` **ProductName** [Subir | Métodos | Propiedades]
Devuelve el nombre del producto como cadena de texto. *Ejemplo:* para `Altova RaptorXML+XBRL Server 2014r2sp1(x64)`, devuelve `Altova RaptorXML+XBRL Server`.

`string` **ProductNameAndVersion** [Subir | Métodos | Propiedades]
Devuelve el nombre y la versión del producto como cadena de texto. *Ejemplo:* para `Altova RaptorXML+XBRL Server 2014r2sp1(x64)`, devuelve `Altova RaptorXML+XBRL Server 2014r2sp1(x64)`.

`bool` **ReportOptionalWarnings** [Subir | Métodos | Propiedades]
Habilita o deshabilita la notificación de advertencias. El valor `true` habilita las advertencias; `false` las deshabilita.

`string` **ServerName** [Subir | Métodos | Propiedades]
Establece el nombre del servidor HTTP. Si ocurre un error, se emite una excepción `RaptorXMLException`.

`string` **ServerPath** [Subir | Métodos | Propiedades]
Especifica en forma de URL la ruta de acceso del servidor HTTP. Si ocurre un error, se emite una excepción `RaptorXMLException`.

`int` **ServerPort** [Subir | Métodos | Propiedades]
Especifica el puerto del servidor HTTP. Es de tipo `ushort`. Si ocurre un error, se emite una excepción `RaptorXMLException`.

`int` **ServicePackVersion** [Subir | Métodos | Propiedades]
Devuelve la versión del service pack del producto como número entero. *Ejemplo:* para `RaptorXML+XBRL Server 2014r2sp1(x64)`, devuelve `1` (del número de versión del service pack `sp1`).

`string` **UserCatalog** [Subir | Métodos | Propiedades]
Indica, en forma de URL, la ubicación del archivo de catálogo definido por el usuario. La cadena indicada debe ser una URL absoluta que dé la ubicación exacta del archivo de catálogo del usuario que se debe usar.

 © 2014 Altova GmbH

IXMLValidator

La interfaz **IXMLValidator** ofrece métodos para probar:

- La validez de un documento XML, DTD o XSD: IsValid. los documentos XML se pueden validar con un DTD o con un esquema XML y las referencia al DTD/esquema pueden estar dentro del documento XML o en el código.
- Si el formato de un documento XML es correcto: IsWellFormed.

Ambos métodos devuelven el valor binario TRUE o FALSE. Las propiedades definen los parámetros de la interfaz.

Métodos
IsValid
IsWellFormed

Propiedades		
AssessmentMode	InputXMLFromText	SchemalocationHints
DTDFileName	LastErrorMessage	SchemaMapping
DTDFromText	PythonScriptFile	SchemaTextArray
EnableNamespaces	SchemaFileArray	Streaming
InputFileArray	SchemaFileName	XincludeSupport
InputTextArray	SchemaFromText	XMLValidationMode
InputXMLFileName	SchemaImports	XSDVersion

Métodos

Los dos métodos de la interfaz **IXMLValidator** son **IsValid** y **IsWellFormed**. Prueban si el documento especificado es válido y si tiene un formato XML correcto respectivamente. Ambos métodos devuelven el valor binario true o false.

---

bool **IsValid(**ENUMValidationType nType**)** [Subir | Métodos | Propiedades]

- Devuelve el resultado de la validación indicada con el valor de ENUMValidationType. Devuelve true si el documento es válido y false si no lo es.
- nType es el valor de ENUMValidationType. El tipo de validación indica si el XML se debe validar con un DTD o un XSD. O si se debe validar un DTD o un XSD. El valor predeterminado es eValidateAny, lo cual indica que RaptorXML debe determinar el tipo

de documento automáticamente.
- Si se produce un error durante la ejecución, se emite una excepción RaptorXMLException. Use la operación LastErrorMessage para obtener más información.

---

bool **IsWellFormed**(ENUMWellformedCheckType nType) [Subir | Métodos | Propiedades]

- Devuelve el resultado de la comprobación de formato XML indicada con el valor de ENUMWellformedCheckType. Devuelve true si el formato es correcto y false si no lo es.
- nType es el valor de ENUMWellformedCheckType. Su valor indica si se debe revisar el formato de un documento XML o de un documento DTD. El valor predeterminado es eWellformedAny.
- Si se produce un error durante la ejecución, se emite una excepción RaptorXMLException. Use la operación LastErrorMessage para obtener más información.

Propiedades

Las propiedades de la interfaz **IXMLValidator** se describen a continuación por orden alfabético. La tabla organiza las propiedades por grupos para facilitar su consulta. Recuerde que las cadenas de entrada que deben interpretarse como direcciones URL deben dar rutas de acceso absolutas. Si se usa una ruta de acceso relativa, debe definir en el módulo de llamada un mecanismo para resolver la ruta de acceso relativa.

Archivos de datos	Archivos de esquema	Procesamiento
InputFileArray	DTDFileName	AssessmentMode
InputTextArray	DTDFromText	EnableNamespaces
InputXMLFileName	SchemaFileArray	LastErrorMessage
InputXMLFromText	SchemaFileName	PythonScriptFile
	SchemaFromText	Streaming
	SchemaImports	XincludeSupport
	SchemalocationHints	XMLValidationMode
	SchemaMapping	XSDVersion
	SchemaTextArray	

ENUMAssessmentMode **AssessmentMode** [Subir | Métodos | Propiedades]

Establece el modo de evaluación del validador XML (strict o lax), según lo indicado por los literales de ENUMAssessmentMode.

---

string **DTDFileName** [Subir | Métodos | Propiedades]
Indica qué documento DTD externo se debe usar para la validación. La cadena indicada debe ser una URL absoluta que dé la ubicación base del DTD que se debe usar.

---

string **DTDFromText** [Subir | Métodos | Propiedades]
Ofrece todo el DTD en forma de cadena de texto.

---

bool **EnableNamespaces** [Subir | Métodos | Propiedades]
Habilita el procesamiento preparado para espacios de nombres. Es útil si quiere buscar errores en la instancia XML derivados del uso incorrecto de espacios de nombres. El valor `true` habilita el procesamiento preparado para espacios de nombres; `false` lo deshabilita. El predeterminado es `false`.

---

object **InputFileArray** [Subir | Métodos | Propiedades]
Ofrece una lista de las direcciones URL de los archivos XML que se deben usar como datos de entrada. La propiedad ofrece un objeto que contiene, como cadenas de texto, las URL absolutas de los archivos XML.

---

object **InputTextArray** [Subir | Métodos | Propiedades]
Ofrece una lista de las direcciones URL de los archivos de texto que se deben usar como datos de entrada. La propiedad ofrece un objeto que contiene, como cadenas de texto, las URL absolutas de los archivos de texto.

---

string **InputXMLFileName** [Subir | Métodos | Propiedades]
Indica qué archivo XML se debe validar. La cadena indicada debe ser una URL absoluta que dé la ubicación base del archivo XML que se debe usar.

---

string **InputXMLFromText** [Subir | Métodos | Propiedades]
Suministra, como cadena de texto, el contenido del documento XML que se debe validar.

---

string **LastErrorMessage** [Subir | Métodos | Propiedades]
Recupera el último mensaje de error del motor de RaptorXML como cadena de texto.

---

string **PythonScriptFile** [Subir | Métodos | Propiedades]

Especifica el archivo de script Python que incluye un procesamiento adicional del archivo XML o XSD indicado para la validación. La cadena indicada debe ser una URL absoluta que dé la ubicación base del script de Python.

---

object **SchemaFileArray** [Subir | Métodos | Propiedades]
Ofrece una lista de las URL de los archivos XSD que se deben usar como esquemas XML externos. La propiedad ofrece un objeto que contiene, como cadenas de texto, las URL absolutas de los archivos de esquema XML.

---

string **SchemaFileName** [Subir | Métodos | Propiedades]
Indica qué esquema XML externo se debe usar para la validación. La cadena indicada debe ser una URL absoluta que dé la ubicación base del archivo de esquema XML que se debe usar.

---

string **SchemaFromText** [Subir | Métodos | Propiedades]
Suministra, como cadena de texto, el contenido del archivo de esquema XML que se debe usar para la validación.

ENUMSchemaImports **SchemaImports** [Subir | Métodos | Propiedades]
Indica qué se debe hacer con las importaciones de esquema, dependiendo del valor de los atributos de los elementos `xs:import`. Esto viene dado por el literal de ENUMSchemaImports seleccionado.

ENUMLoadSchemalocation **SchemalocationHints** [Subir | Métodos | Propiedades]
Especifica el mecanismo que se debe usar para encontrar el esquema. El mecanismo viene dado por el literal de ENUMLoadSchemalocation seleccionado.

ENUMSchemaMapping **SchemaMapping** [Subir | Métodos | Propiedades]
Establece qué asignación se usa para encontrar el esquema. La asignación viene dada por el literal de ENUMSchemaMapping seleccionado.

---

object **SchemaTextArray** [Subir | Métodos | Propiedades]
Ofrece un conjunto de cadenas de texto, que son los archivos XSD que se deben usar como esquemas XML externos. La propiedad ofrece un objeto que contiene, como cadenas de texto, la cadena de texto correspondiente a cada archivo XSD.

---

bool **Streaming** [Subir | Métodos | Propiedades]
Habilita la validación por transmisión de secuencias de datos. En el modo de transmisión por secuencias, los datos almacenados en memoria se minimizan y el procesamiento es más rápido. El valor `true` habilita la validación por transmisión de secuencias; `false` la deshabilita. El valor

 © 2014 Altova GmbH

predeterminado es `true`.

`bool` **XincludeSupport** [Subir | Métodos | Propiedades]
Habilita el uso de elementos `XInclude`. El valor `true` habilita los elementos XInclude; `false` los deshabilita. El valor predeterminado es `false`.

ENUMXMLValidationMode **XMLValidationMode** [Subir | Métodos | Propiedades]
Establece el modo de validación XML (validación o comprobación de formato XML). El modo es el indicado por el literal de ENUMXMLValidationMode.

ENUMXSDVersion **XSDVersion** [Subir | Métodos | Propiedades]
Indica la versión de XML Schema que se debe usar para validar el documento XML. Sus valores son los literales de ENUMXSDVersion.

IXSLT

La interfaz **IXSLT** ofrece métodos y propiedades para ejecutar una transformación XSLT 1.0, 2.0 o 3.0. Los resultados se pueden guardar en un archivo o devolverse como cadena de texto. La interfaz también permite pasar parámetros XSLT a la hoja de estilos XSLT. Las direcciones URL de los archivo XML y XSLT se pueden dar como cadenas de texto a través de las propiedades de la interfaz. Otra opción es construir documentos XML y XSLT dentro del código como cadenas de texto.

Nota: Cuando las cadenas de texto de entrada se deben interpretar como URL, deben utilizarse rutas de acceso absolutas. Si se usa una ruta de acceso relativa, debe definir en el módulo de llamada un mecanismo para resolver la ruta de acceso relativa.

Nota: Puede usar el motor XSLT 2.0 o 3.0 de RaptorXML en el modo compatible con versiones anteriores para procesar hojas de estilos XSLT 1.0. No obstante, el resultado puede ser diferente al que se obtendría con el motor XSLT 1.0.

Métodos
IsValid
Execute
ExecuteAndGetResultAsString
ExecuteAndGetResultAsStringWithBaseOutputURI
AddExternalParameter
ClearExternalParameterList

Propiedades

ChartExtensionsEnabled	JavaBarcodeExtensionLocation	SchemaMapping
DotNetExtensionsEnabled	JavaExtensionsEnabled	StreamingSerialization
EngineVersion	LastErrorMessage	XincludeSupport
IndentCharacters	LoadXMLWithPSVI	XMLValidationMode
InitialTemplateMode	NamedTemplateEntryPoint	XSDVersion
InputXMLFileName	SchemaImports	XSLFileName
InputXMLFromText	SchemalocationHints	XSLFromText

Métodos

Los métodos de la **IXSLT** se describen a continuación. Recuerde que las cadenas de entrada que deben interpretarse como direcciones URL deben dar rutas de acceso absolutas. Si se usa una ruta de acceso relativa, debe definir en el módulo de llamada un mecanismo para resolver la ruta de acceso relativa.

Métodos
IsValid
Execute
ExecuteAndGetResultAsString
ExecuteAndGetResultAsStringWithBaseOutputURI
AddExternalParameter
ClearExternalParameterList

bool **IsValid()** [Subir | Métodos | Propiedades]

- Devuelve el resultado de validar la hoja de estilos XSLT de acuerdo con la especificación XSLT indicada en ENUMXSLTVersion (ver la propiedad EngineVersion). El resultado es true si el XSLT es válido y false si no lo es.
- Si ocurre un error, se emite una excepción RaptorXMLException. Use la operación LastErrorMessage para obtener más información.

---

bool **Execute(string bstrResultFileName)** [Subir | Métodos | Propiedades]

- Ejecuta la transformación XSLT de acuerdo con la especificación XSLT indicada en ENUMXSLTVersion (ver la propiedad EngineVersion) y guarda el resultado en un archivo de salida.
- El archivo de salida viene dado por bstrResultFileName, que es una cadena que indica

la URL del archivo de salida.
- El resultado es `true` si la transformación finaliza correctamente y `false` si se producen errores.
- Si ocurre un error, se emite una excepción `RaptorXMLException`. Use la operación `LastErrorMessage` para obtener más información.

`string` **`ExecuteAndGetResultAsString()`** [Subir | Métodos | Propiedades]

- Ejecuta la transformación XSLT de acuerdo con la especificación indicada en `ENUMXSLTVersion` (ver la propiedad `EngineVersion`) y devuelve el resultado de la transformación como cadena de texto.
- Si ocurre un error durante la transformación, se emite una excepción `RaptorXMLException`. Use la operación `LastErrorMessage` para obtener más información.

`string` **`ExecuteAndGetResultAsStringWithBaseOutputURI(`**`string bstrBaseURI`**`)`** [Subir | Métodos | Propiedades]

- Ejecuta la transformación XSLT de acuerdo con la especificación indicada en `ENUMXSLTVersion` (ver la propiedad `EngineVersion`) y devuelve el resultado de la transformación como cadena de texto en la ubicación definida por el URI base (la cadena `bstrBaseURI`).
- Si ocurre un error durante la transformación, se emite una excepción `RaptorXMLException`. Use la operación `LastErrorMessage` para obtener más información.

`void` **`AddExternalParameter(`**`string bstrName, string bstrValue`**`)`** [Subir | Métodos | Propiedades]

- Añade el nombre y el valor de un parámetro externo: `bstrName` y `bstrValue` son cadenas.
- Cada parámetro externo y su valor se puede especificar en una llamada distinta al método. Los parámetros se deben declarar en el documento XSLT y si quiere puede añadir una declaración de tipo. Sea cual sea la declaración de tipo del documento XSLT, no es necesario ningún delimitador concreto si el valor del parámetro se indica con `AddExternalParameter`.

`void` **`ClearExternalParameterList()`** [Subir | Métodos | Propiedades]

- Borra la lista de parámetros externos creada con el método `AddExternalParameter`.

Propiedades
Las propiedades de la interfaz **IXSLT** se describen a continuación por orden alfabético. La tabla organiza las propiedades por grupos para facilitar su consulta. Recuerde que las cadenas de entrada que deben interpretarse como direcciones URL deben dar rutas de acceso absolutas. Si se usa una ruta de acceso relativa, debe definir en el módulo de llamada un mecanismo para resolver la ruta de acceso relativa.

XML	XSLT	Schema
InputXMLFileName	EngineVersion	SchemaImports
InputXMLFromText	XSLFileName	SchemalocationHints
LoadXMLWithPSVI	XSLFromText	SchemaMapping
XincludeSupport		XSDVersion
XMLValidationMode		

Procesamiento	Extensiones
IndentCharacters	ChartExtensionsEnabled
InitialTemplateMode	DotNetExtensionsEnabled
LastErrorMessage	JavaBarcodeExtensionLocation
NamedTemplateEntryPoint	JavaExtensionsEnabled
StreamingSerialization	

bool **ChartExtensionsEnabled** [Subir | Métodos | Propiedades]
Habilita o deshabilita las funciones de extensión de Altova para gráficos. El valor `true` habilita las extensiones para gráficos; `false` las deshabilita. El valor predeterminado es `true`.

---

bool **DotNetExtensionsEnabled** [Subir | Métodos | Propiedades]
Habilita o deshabilita las funciones de extensión de Visual Studio .NET. El valor `true` habilita las extensiones .NET; `false` las deshabilita. El valor predeterminado es `true`.

ENUMXSLTVersion **EngineVersion** [Subir | Métodos | Propiedades]
Indica qué versión XSLT se debe usar (1.0, 2.0 o 3.0). El valor de la propiedad es un literal de ENUMXSLTVersion.

---

string **IndentCharacters** [Subir | Métodos | Propiedades]

 © 2014 Altova GmbH

Establece el carácter que debe utilizarse para la sangría.

---

string **InitialTemplateMode** [Subir | Métodos | Propiedades]
Establece el modo inicial para el procesamiento XSLT. Se procesarán aquellas plantillas cuyo valor de modo sea igual a la cadena indicada.

---

string **InputXMLFileName** [Subir | Métodos | Propiedades]
Especifica la ubicación del archivo XML que se debe transformar. La cadena indicada debe ser una URL absoluta que dé la ubicación exacta del archivo XML que se debe usar.

---

string **InputXMLFromText** [Subir | Métodos | Propiedades]
Suministra, como cadena de texto, el contenido del archivo XML que se debe transformar.

---

string **JavaBarcodeExtensionLocation** [Subir | Métodos | Propiedades]
Especifica la ubicación del archivo de extensiones para códigos de barras. Consulte este apartado para obtener más información. La cadena indicada debe ser una URL absoluta que dé la ubicación base del archivo que se debe usar.

---

bool **JavaExtensionsEnabled** [Subir | Métodos | Propiedades]
Habilita o deshabilita las extensiones Java. El valor `true` habilita las extensiones Java; `false` las deshabilita. El valor predeterminado es `true`.

---

string **LastErrorMessage** [Subir | Métodos | Propiedades]
Recupera el último mensaje de error del motor de RaptorXML como cadena de texto.

---

bool **LoadXMLWithPSVI** [Subir | Métodos | Propiedades]
Habilita o deshabilita la opción de carga y uso del conjunto Post Schema Validation Infoset (PSVI). Si se carga el PSVI, la información obtenida del esquema se puede usar para calificar datos del documento XML. El valor `true` habilita la carga de PSVI; `false` la deshabilita.

---

string **NamedTemplateEntryPoint** [Subir | Métodos | Propiedades]
Especifica el nombre, como cadena de texto, de la plantilla con nombre que debe utilizarse como punto de entrada de la transformación.

ENUMSchemaImports **SchemaImports** [Subir | Métodos | Propiedades]
Indica qué se debe hacer con las importaciones de esquemas, dependiendo del valor de los atributos de los elementos `xs:import`. Esto viene dado por el literal de ENUMSchemaImports

seleccionado.

ENUMLoadSchemalocation **SchemalocationHints** [Subir | Métodos | Propiedades]
Especifica el mecanismo que se debe usar para encontrar el esquema. El mecanismo viene dado por el literal de ENUMLoadSchemalocation seleccionado.

ENUMSchemaMapping **SchemaMapping** [Subir | Métodos | Propiedades]
Establece qué asignación se usa para encontrar el esquema. La asignación viene dada por el literal de ENUMSchemaMapping seleccionado.

---

bool **StreamingSerialization** [Subir | Métodos | Propiedades]
Habilita la serialización de secuencias de datos. En el modo de transmisión por secuencias, los datos almacenados en memoria se minimizan y el procesamiento es más rápido. El valor `true` habilita la serialización de secuencias de datos; `false` la deshabilita.

---

bool **XincludeSupport** [Subir | Métodos | Propiedades]
Habilita el uso de elementos `XInclude`. El valor `true` habilita el uso de XInclude; `false` lo deshabilita. El valor predeterminado es `false`.

ENUMXMLValidationMode **XMLValidationMode** [Subir | Métodos | Propiedades]
Establece el modo de validación XML (validación o comprobación de formato XML). El modo es el indicado por el literal de ENUMXMLValidationMode.

ENUMXSDVersion **XSDVersion** [Subir | Métodos | Propiedades]
Indica la versión de XML Schema que se debe usar para validar el documento XML. Sus valores son los literales de ENUMXSDVersion.

---

string **XSLFileName** [Subir | Métodos | Propiedades]
Especifica qué archivo XSLT se debe usar para la transformación. La cadena indicada debe ser una URL absoluta que dé la ubicación del archivo XSLT que se debe usar.

---

string **XSLFromText** [Subir | Métodos | Propiedades]
Ofrece, en forma de cadena de texto, el contenido del documento XSLT que se debe usar para la transformación.

 © 2014 Altova GmbH

IXQuery

La interfaz **IXQuery** ofrece métodos y propiedades para ejecutar un documento XQuery 1.0 o XQuery 3.0. Los resultados se pueden guardar en un archivo o devolverse como cadena de texto. La interfaz también permite pasar variables XQuery externas al documento XQuery. Las URL de los archivos XQuery y XML se pueden dar como cadenas de texto a través de las propiedades de la interfaz. Otra opción es construir los documentos XML y XQuery dentro del código como cadenas de texto.

Note: Cuando las cadenas de texto de entrada se deben interpretar como URL, deben utilizarse rutas de acceso absolutas. Si se usa una ruta de acceso relativa, debe definir en el módulo de llamada un mecanismo para resolver la ruta de acceso relativa.

Métodos
IsValid
Execute
ExecuteAndGetResultAsString
AddExternalVariable
ClearExternalParameterList

Propiedades			
ChartExtensionsEnabled	InputXMLFromText	OutputEncoding	XMLValidationMode
DotNetExtensionsEnabled	JavaBarcodeExtensionLocation	OutputIndent	XQueryFileName
EngineVersion	JavaExtensionsEnabled	OutputMethod	XQueryFromText
IndentCharacters	LastErrorMessage	OutputOmitXMLDeclaration	XSDVersion
InputXMLFileName	LoadXMLWithPSVI	XincludeSupport	

Métodos

Los métodos de la **IXQuery** se describen a continuación. Recuerde que las cadenas de entrada que deben interpretarse como direcciones URL deben dar rutas de acceso absolutas. Si se usa una ruta de acceso relativa, debe definir en el módulo de llamada un mecanismo para resolver la ruta de acceso relativa.

Métodos
IsValid

Execute
ExecuteAndGetResultAsString
AddExternalVariable
ClearExternalParameterList

bool **IsValid()** [Subir | Métodos | Propiedades]

- Devuelve el resultado de validar el documento XQuery de acuerdo con la especificación XQuery indicada en ENUMXQueryVersion (ver la propiedad EngineVersion). El resultado es true si el documento es válido y false si no lo es.
- Si ocurre un error, se emite una excepción RaptorXMLException. Use la operación LastErrorMessage para obtener más información.

---

bool **Execute(string bstrOutputFile)** [Subir | Métodos | Propiedades]

- Ejecuta el documento XQuery de acuerdo con la especificación XQuery indicada en ENUMXQueryVersion (ver la propiedad EngineVersion), y guarda el resultado en un archivo de salida.
- El archivo de salida viene dado por bstrOutputFile, que es una cadena que indica la URL del archivo de salida.
- El valor binario true se devuelve si la operación finaliza correctamente y false si hay errores.
- Si ocurre un error durante la transformación, se emite una excepción RaptorXMLException. Use la operación LastErrorMessage para obtener más información.

---

string **ExecuteAndGetResultAsString()** [Subir | Métodos | Propiedades]

- Ejecuta la transformación XQuery de acuerdo con la especificación XQuery indicada en ENUMXQueryVersion (ver la propiedad EngineVersion) y devuelve el resultado de la transformación como cadena de texto.
- Si ocurre un error durante la transformación, se emite una excepción RaptorXMLException. Use la operación LastErrorMessage para obtener más información.

---

void **AddExternalVariable(**string bstrName, string bstrValue**)** [Subir | Métodos | Propiedades]

- Añade el nombre y valor de una variable externa: bstrName y bstrValue son cadenas de texto.
- Cada variable externa y su valor deben indicarse en una llamada distinta al método. Las

 © 2014 Altova GmbH

variables deben declararse en el documento XQuery y si quiere puede incluir una declaración de tipo. Sea cual sea la declaración de tipo del documento XQuery, no es necesario incluir ningún delimitador especial si el valor de la variable se indica por medio de `AddExternalVariable`.

`void` **ClearExternalVariableList()** [Subir | Métodos | Propiedades]

- Borra la lista de variables externas creadas con el método `AddExternalVariable`.

Propiedades

Las propiedades de la interfaz **`IXQuery`** se describen a continuación por orden alfabético. La tabla organiza las propiedades por grupos para facilitar su consulta. Recuerde que las cadenas de entrada que deben interpretarse como direcciones URL deben dar rutas de acceso absolutas. Si se usa una ruta de acceso relativa, debe definir en el módulo de llamada un mecanismo para resolver la ruta de acceso relativa.

XML	XQuery	Procesamiento	Extensiones
InputXMLFileName	EngineVersion	IndentCharacters	ChartExtensionsEnabled
InputXMLFromText	XQueryFileName	LastErrorMessage	DotNetExtensionsEnabled
LoadXMLWithPSVI	XQueryFromText	OutputEncoding	JavaBarcodeExtensionLocation
XincludeSupport		OutputIndent	JavaExtensionsEnabled
XMLValidationMode		OutputMethod	
XSDVersion		OutputOmitXMLDeclaration	

`bool` **ChartExtensionsEnabled** [Subir | Métodos | Propiedades]

Habilita o deshabilita las funciones de extensión de Altova para gráficos. El valor `true` habilita las extensiones para gráficos; `false` las deshabilita. El valor predeterminado es `true`.

`bool` **DotNetExtensionsEnabled** [Subir | Métodos | Propiedades]

Habilita o deshabilita las funciones de extensión Visual Studio .NET. El valor `true` habilita las extensiones .NET; `false` las deshabilita. El valor predeterminado es `true`.

ENUMXQueryVersion **EngineVersion** [Subir | Métodos | Propiedades]
Indica qué versión de XQuery se debe usar (1.0 o 3.0). El valor de la propiedad es un literal de ENUMXQueryVersion.

---

string **IndentCharacters** [Subir | Métodos | Propiedades]
Establece el carácter que debe utilizarse para la sangría.

---

string **InputXMLFileName** [Subir | Métodos | Propiedades]
Indica la ubicación del archivo XML que se debe procesar. La cadena indicada debe ser una URL absoluta que dé la ubicación exacta del archivo XML que se debe usar.

---

string **InputXMLFromText** [Subir | Métodos | Propiedades]
Suministra, como cadena de texto, el contenido del archivo XML que se debe procesar.

---

string **JavaBarcodeExtensionLocation** [Subir | Métodos | Propiedades]
Especifica la ubicación del archivo de extensión para códigos de barras. Para más información consulte este apartado. La cadena indicada debe ser una URL absoluta que dé la ubicación base del archivo que se debe usar.

---

bool **JavaExtensionsEnabled** [Subir | Métodos | Propiedades]
Habilita o deshabilita las extensiones Java. El valor `true` habilita las extensiones Java; `false` las deshabilita. El valor predeterminado es `true`.

---

string **LastErrorMessage** [Subir | Métodos | Propiedades]
Recupera el último mensaje de error del motor de RaptorXML como cadena de texto.

---

bool **LoadXMLWithPSVI** [Subir | Métodos | Propiedades]
Habilita o deshabilita la opción de carga y uso del conjunto Post Schema Validation Infoset (PSVI). Si se carga el PSVI, la información obtenida del esquema se puede usar para calificar datos del documento XML. El valor `true` habilita la carga de PSVI; `false` la deshabilita.

---

string **OutputEncoding** [Subir | Métodos | Propiedades]
Establece la codificación del documento de salida. Use un nombre de codificación IANA oficial (p. ej. `UTF-8`, `UTF-16`, `US-ASCII`, `ISO-8859-1`) como cadena de texto.

---

bool **OutputIndent** [Subir | Métodos | Propiedades]

 © 2014 Altova GmbH

Habilita o deshabilita la sangría en el documento de salida. El valor `true` habilita la sangría; `false` la deshabilita.

`string` **OutputMethod** [Subir | Métodos | Propiedades]
Especifica la serialización del documento de salida. Los valores válidos son: `xml` | `xhtml` | `html` | `text`. El valor predeterminado es `xml`.

`bool` **OutputOmitXMLDeclaration** [Subir | Métodos | Propiedades]
Habilita o deshabilita la inclusión de la declaración XML en el documento de salida. El valor `true` omite la declaración; `false` la incluye. El valor predeterminado es `false`.

`bool` **XincludeSupport** [Subir | Métodos | Propiedades]
Habilita o deshabilita el uso de elementos `XInclude`. El valor `true` habilita el uso de XInclude; `false` lo deshabilita. El valor predeterminado es `false`.

`ENUMXMLValidationMode` **XMLValidationMode** [Subir | Métodos | Propiedades]
Establece el modo de validación XML (validación o comprobación de formato XML). El modo es el indicado por el literal de `ENUMXMLValidationMode`.

`string` **XQueryFileName** [Subir | Métodos | Propiedades]
Indica qué archivo XQuery se debe usar. La cadena indicada debe ser una URL absoluta que dé la ubicación del archivo XQuery que se debe usar.

`string` **XQueryFromText** [Subir | Métodos | Propiedades]
Suministra, como cadena de texto, el contenido del archivo XQuery que se debe usar.

`ENUMXSDVersion` **XSDVersion** [Subir | Métodos | Propiedades]
Indica la versión de XML Schema que se debe usar para validar el documento XML. Sus valores son los literales de `ENUMXSDVersion`.

IXBRL

La interfaz **IXBRL** ofrece métodos para validar documentos de instancia XBRL y taxonomías XBRL, así como fórmulas. Los resultados son los valores booleanos `true` o `false`. La interfaz también permite pasar parámetros de fórmula para la evaluación de las mismas. Las aserciones de fórmulas y el resultado también se puede consultar y se pueden devolver como cadenas de texto. Las propiedades definen los parámetros de la interfaz.

Note: Cuando las cadenas de texto de entrada se deben interpretar como URL, deben utilizarse rutas de acceso absolutas. Si se usa una ruta de acceso relativa, debe definir en el módulo de llamada un mecanismo para resolver la ruta de acceso relativa.

Estructuras

Se define la siguiente estructura.

```
public struct XBRLParamValuePair
{
    String ParamType;
    String ParamValue;
};
```

Métodos

Los métodos de la **IXBRL** se describen a continuación. Cuando las cadenas de texto de entrada se deben interpretar como URL, deben utilizarse rutas de acceso absolutas. Si se usa una ruta de acceso relativa, debe definir en el módulo de llamada un mecanismo para resolver la ruta de acceso relativa.

`bool` **IsValid(**ENUMXBRLValidationType `nType`**)** [Subir | Métodos | Propiedades]

- Devuelve el resultado de validar el documento de instancia XBRL o la taxonomía.
- `nType` es el valor de ENUMXBRLValidationType. El tipo de validación indica si lo que se valida es un documento XBRL de instancia o una taxonomía XBRL. El valor predeterminado es `eValidateXBRLAny`, que indica que RaptorXML debe determinar el tipo de documento automáticamente
- Si se produce un error durante la validación, se emite una excepción `RaptorXMLException`. Use la operación LastErrorMessage para obtener más información.

`bool` **EvaluateFormula()** [Subir | Métodos | Propiedades]

- Evalúa las fórmulas XBRL de un documento XBRL de instancia. Devuelve `true` si son válidas y `false` si alguna fórmula no es válida.
- Si se produce un error durante la evaluación, se emite una excepción `RaptorXMLException`. Use la operación LastErrorMessage para obtener más información.

 © 2014 Altova GmbH

`bool` **`GenerateTables()`** [Subir | Métodos | Propiedades]

- Genera la tabla XBRL en un documento XBRL de instancia. Devuelve `true` si la generación es correcta y `false` si no lo es.
- Si se produce un error durante la ejecución, se emite una excepción `RaptorXMLException`. Use la operación `LastErrorMessage` para obtener más información.

`void` **`AddFormulaArrayParameter(`**`string sDefaultType, string sName, object[] variantValues`**`)`** [Subir | Métodos | Propiedades]

- Añade un parámetro de matriz utilizado en el proceso de evaluación de fórmulas.
- Todos los argumentos son cadenas: `sDefaultType` es el tipo de datos predeterminado de valores de no pares dentro de los valores de matriz. El tipo predeterminado es `xs:string`; `sName` es el nombre de parámetro; `variantValues` es una matriz de valor y pares valor/tipo de datos.
- Para más información y ejemplos de código consulte el apartado Parámetros de fórmulas XBRL.

`void` **`AddFormulaParameter(`**`string sType, string sName, string sValue, string sNamespace`**`)`** [Subir | Métodos | Propiedades]

- Añade un parámetro para la evaluación de fórmulas. Es obsoleto.
- Todos los argumentos son cadenas: `sType` es el tipo de datos del parámetro; `sName` es el nombre del parámetro; `sValue` es el valor del parámetro; y `sNamespace` es el espacio de nombres del parámetro.
- Cada parámetro se debe especificar en una llamada al método distinta.

`void` **`AddFormulaParameter(`**`string sType, string sName, string sValue, string sNamespace = ""`**`)`** [Subir | Métodos | Propiedades]

- Añade un parámetro para la evaluación de fórmulas.
- Todos los argumentos son cadenas: `sType` es el tipo de datos del parámetro; `sName` es el nombre del parámetro; `sValue` es el valor del parámetro; y `sNamespace` es el espacio de nombres del parámetro y es la cadena vacía.
- Cada parámetro se debe especificar en una llamada al método distinta.

`void` **`AddFormulaParameterNamespace(`**`string sPrefix, string sURI`**`)` [Subir | Métodos | Propiedades]

- Define un espacio de nombres utilizado en los QName de los nombres, tipos y valores de parámetro.
- Todos los argumentos son cadenas: `sPrefix` es el prefijo de espacio de nombres de los valores pasados a `AddFormulaArrayParameter`; `sURI` es el URI de espacio de nombres.
- Cada parámetro se debe especificar en una llamada al método distinta.

---

void **ClearFormulaParameterList()** [Subir | Métodos | Propiedades]

- Borra la lista de parámetros de fórmula creados con el método AddFormulaParameter.

---

string **ReadFormulaAssertions()** [Subir | Métodos | Propiedades]

- Lee las aserciones de fórmulas del archivo que se está evaluando.

---

string **ReadFormulaOutput()** [Subir | Métodos | Propiedades]

- Lee el resultado de las aserciones de fórmulas del archivo.

Propiedades

Las propiedades de la interfaz **IXBRL** se describen a continuación por orden alfabético. La tabla organiza las propiedades por grupos para facilitar su consulta. Recuerde que las cadenas de entrada que deben interpretarse como direcciones URL deben dar rutas de acceso absolutas. Si se usa una ruta de acceso relativa, debe definir en el módulo de llamada un mecanismo para resolver la ruta de acceso relativa.

string **AddAssertionForProcessing** [Subir | Métodos | Propiedades]

Limita la evaluación de aserciones a la aserción dada. Llámele varias veces para especificar más de una aserción. Use `##none` para no evaluar ninguna aserción y `##all` para evaluar todas.

---

string **AddAssertionSetForProcessing** [Subir | Métodos | Propiedades]

Limita la evaluación de conjuntos de aserciones al conjunto de aserciones dado. Llámele varias veces para especificar más de un conjunto de aserciones. Use `##none` para no evaluar ningún conjunto de aserciones y `##all` para evaluar todos.

 © 2014 Altova GmbH

`string` **AddTableForProcessing** [Subir | Métodos | Propiedades]
Limita la generación de tablas a la tabla dada. Llámele varias veces para especificar más de una tabla. Use `##none` para no generar ninguna tabla y `##all` para generar todas.

`string` **ConceptLabelLinkrole** [Subir | Métodos | Propiedades]
Especifica la función de vínculo extendido preferida a la hora de representar etiquetas de concepto.

`string` **ConceptLabelRole** [Subir | Métodos | Propiedades]
Especifica la función de etiqueta preferida a la hora de representar etiquetas de concepto. Predeterminada: `http://www.xbrl.org/2008/role/label`.

`bool` **DimensionExtensionEnabled** [Subir | Métodos | Propiedades]
Habilita o deshabilita la validación con la extensión XBRL Dimension. El valor `true` habilita la validación la extensión XBRL Dimension; `false` la deshabilita. El valor predeterminado es `true`.

`bool` **FormulaAssertionsAsXML** [Subir | Métodos | Propiedades]
Habilita el formato XML del archivo de aserciones de fórmulas cuando RaptorXML se ejecuta con las aserciones habilitadas. El valor `true` habilita el formato XML; el valor `false` genera resultados en formato JSON. El valor predeterminado es `false`.

`string` **FormulaAssertionsOutput** [Subir | Métodos | Propiedades]
Indica la ubicación del archivo de salida de las aserciones de fórmulas. Es obligatorio indicar una ruta de acceso completa.

`bool` **FormulaExtensionEnabled** [Subir | Métodos | Propiedades]
Habilita o deshabilita la validación con la extensión XBRL Formula. El valor `true` habilita la validación la extensión XBRL Formula; `false` la deshabilita. El valor predeterminado es `true`.

`string` **FormulaOutput** [Subir | Métodos | Propiedades]
Indica la ubicación del resultado del archivo de evaluación de fórmulas XBRL. Es obligatorio indicar una ruta de acceso completa.

`string` **FormulaParameterFile** [Subir | Métodos | Propiedades]
Indica la ubicación del archivo de parámetros de fórmulas. Es obligatorio indicar una ruta de acceso completa.

---

bool **FormulaPreloadSchemas** [Subir | Métodos | Propiedades]
Define si se cargan o no previamente los esquemas de XBRL Formula. El valor `true` habilita la carga previa de estos esquemas. El valor predeterminado es `false`, que no carga los esquemas previamente.

---

string **GenericLabelLinkrole** [Top | Methods | Properties]
Especifica la función de vínculo extendido preferida a la hora de representar etiquetas genéricas.

---

string **GenericLabelRole** [Top | Methods | Properties]
Especifica la función de etiqueta preferida a la hora de representar etiquetas genéricas. Predeterminada: `http://www.xbrl.org/2008/role/label`.

---

object **InputFileArray** [Subir | Métodos | Propiedades]
Establece el conjunto de archivos XBRL que se usarán como datos de entrada/instancias. Este conjunto es un objeto que contiene las cadenas de las URL absolutas de los archivos de entrada.

---

string **InputFileName** [Subir | Métodos | Propiedades]
Indica el nombre de archivo y la ubicación del archivo de instancia XBRL. La cadena dada debe ser una URL absoluta. Las rutas de acceso relativas se pueden resolver en relación a una ubicación base, dependiendo del mecanismo definido en el módulo de llamada.

---

string **InputFromText** [Subir | Métodos | Propiedades]
Suministra el contenido del documento XBRL de entrada como texto.

---

object **InputTextArray** [Subir | Métodos | Propiedades]
Establece el conjunto de archivos de texto que se usarán como datos de entrada. Este conjunto es un objeto que contiene las cadenas de las URL absolutas de los archivos de entrada.

---

string **LabelLang** [Top | Methods | Properties]
Especifica el idioma de etiqueta preferido a la hora de representar etiquetas. Predeterminado: `en`.

---

string **LastErrorMessage** [Subir | Métodos | Propiedades]
Recupera el último mensaje de error del motor de RaptorXML como cadena de texto.

---

bool **ParallelAssessment** [Top | Methods | Properties]

 © 2014 Altova GmbH

Habilita/deshabilita la evaluación de la validez de esquemas en paralelo.

---

bool **PreloadSchemas** [Subir | Métodos | Propiedades]
Define si se cargan previamente los esquemas XBRL 2.1 o no. El valor `true` carga estos esquemas previamente. El valor predeterminado es `true`.

---

string **PythonScriptFile** [Subir | Métodos | Propiedades]
Indica qué archivo de script Python ofrece procesamiento adicional para el archivo XML o XSD sometido a validación. La cadena indicada debe ser una URL absoluta que dé la ubicación base del archivo de script Python.

ENUMSchemaImports **SchemaImports** [Subir | Métodos | Propiedades]
Indica qué se debe hacer con las importaciones de esquema, dependiendo del valor de los atributos de los elementos `xs:import`. Esto viene dado por el literal de ENUMSchemaImports seleccionado.

ENUMLoadSchemalocation **SchemalocationHints** [Subir | Métodos | Propiedades]
Especifica el mecanismo que se debe usar para encontrar el esquema. El mecanismo viene dado por el literal de ENUMLoadSchemalocation seleccionado.

ENUMSchemaMapping **SchemaMapping** [Subir | Métodos | Propiedades]
Establece qué asignación se usa para encontrar el esquema. La asignación viene dada por el literal de ENUMSchemaMapping seleccionado.

---

bool **TableEliminateEmptyRows** [Top | Methods | Properties]
Habilita la eliminación de filas/columnas vacías en la tabla generada en formato de salida HTML.

---

bool **TableExtensionEnabled** [Top | Methods | Properties]
Habilita/deshabilita la extensión XBRL Table 1.0.

---

string **TableOutput** [Top | Methods | Properties]
Especifica el nombre de archivo y la ubicación del archivo de salida de la tabla generada. La cadena dada debe ser la ruta de acceso completa del archivo de salida.

ENUMTableOutputFormat **TableOutputFormat** [Top | Methods | Properties]
Especifica el formato del archivo de salida de generación de tablas.

---

bool **TablePreloadSchemas** [Top | Methods | Properties]
Habilita/deshabilita la carga previa de los esquemas de la especificación XBRL Table 1.0.

---

bool **TreatXBRLInconsistenciesAsErrors** [Subir | Métodos | Propiedades]
El valor `true` detiene la validación XBRL y define el archivo como no válido si se encuentran en él incoherencias de acuerdo con la especificación XBRL 2.1. El valor predeterminado es `false`: las incoherencias XBRL definidas en la especificación XBRL 2.1 no se consideran errores.

---

bool **XincludeSupport** [Subir | Métodos | Propiedades]
Habilita el uso de los elementos `XInclude`. El valor `true` habilita el uso de XInclude; `false` lo deshabilita.

 © 2014 Altova GmbH

7.4.2 Enumeraciones

Las enumeraciones definidas son estas:

ENUMAssessmentMode

ENUMErrorFormat

ENUMLoadSchemalocation

ENUMQueryVersion

ENUMSchemaImports

ENUMSchemaMapping

ENUMValidationType

ENUMWellformedCheckType

ENUMXBRLValidationType

ENUMXMLValidationMode

ENUMXQueryVersion

ENUMXSDVersion

ENUMXSLTVersion

ENUMAssessmentMode

Descripción

Contiene literales de enumeración que definen el modo de evaluación del validador XML: `Strict` o `Lax`.

Utilizada por

Interfaz	*Operación*
IXMLValidator	AssessmentMode

Literales de la enumeración

```
eAssessmentModeStrict        = 0
eAssessmentModeLax           = 1
```

`eAssessmentModeStrict`

Establece el modo de evaluación de la validez del esquema en `Strict`. Es el valor predeterminado.

eAssessmentModeLax
Establece el modo de evaluación de la validez del esquema en `Lax`.

ENUMErrorFormat

Descripción
Contiene literales de enumeración que especifican el formato de los errores de salida.

Utilizada por

Interfaz	*Operación*
IServer	ErrorFormat

Literales de la enumeración

```
eFormatText          = 0
eFormatShortXML      = 1
eFormatLongXML       = 2
```

eFormatText
Establece el formato de los errores de salida en `Text`. Es el valor predeterminado.

eFormatShortXML
Establece el formato de los errores de salida en `ShortXML`. Este formato es una versión abreviada del formato `LongXML`.

eFormatLongXML
Establece el formato de los errores de salida en `LongXML`. Este formato es el que ofrece información más detallada.

ENUMLoadSchemaLocation

Descripción
Contiene literales de enumeración que indican cómo determinar la ubicación del esquema.

Utilizada por

 © 2014 Altova GmbH

Interfaz	*Operación*
IXBRL	SchemalocationHints
IXMLValidator	SchemalocationHints
IXSLT	SchemalocationHints

Literales de la enumeración

```
eSHLoadBySchemalocation          = 0
eSHLoadByNamespace               = 1
eSHLoadCombiningBoth             = 2
eSHLoadIgnore                    = 3
```

eSHLoadBySchemalocation
Asigna a Load Schemalocation el valor `LoadBySchemalocation`. Usa la URL de schema location de los atributos `xsi:schemaLocation` y `xsi:noNamespaceSchemaLocation` de los documentos XML o XBRL de instancia. Es el **valor predeterminado**.

eSHLoadByNamespace
Asigna a Load Schemalocation el valor `LoadByNamespace`. Use la parte de espacio de nombres del atributo `xsi:schemaLocation` (en el caso de `xsi:noNamespaceSchemaLocation` es una cadena vacía) y busca el esquema a través de la asignación de catálogo.

eSHLoadCombiningBoth
Asigna a Load Schemalocation el valor `CombiningBoth`. Si el espacio de nombres o la URL tienen una asignación de catálogo, esta asignación de catálogo se utiliza. Si ambas tienen una asignación de catálogo, entonces es el valor del parámetro `ENUMSchemaMapping` lo que decide cuál de las asignaciones se utiliza. Si ninguna tiene una asignación de catálogo, se usa la URL.

eSHLoadIgnore
Asigna a Load Schemalocation el valor `LoadIgnore`. Si el valor del parámetro es `eSHLoadIgnore`, se ignoran los atributos `xsi:schemaLocation` y `xsi:noNamespaceSchemaLocation`.

ENUMQueryVersion

Descripción
Contiene literales de enumeración que especifican la versión XQuery que se debe usar: XQuery 1.0 o 3.0.

Literales de la enumeración

```
eXQVersion10        = 1
eXQVersion30        = 3
```

eXQVersion10
Establece que la versión XQuery es XQuery 1.0.

eXQVersion30
Establece que la versión XQuery es XQuery 3.0.

ENUMSchemaImports

Descripción
Contiene los literales de enumeración que definen el comportamiento de los elementos `xs:import`. El elemento `xs:import` tiene los atributos opcionales `namespace` y `schemaLocation`.

Utilizada por

Interfaz	*Operación*
IXBRL	SchemaImports
IXMLValidator	SchemaImports
IXSLT	SchemaImports

Literales de la enumeración

```
eSILoadBySchemalocation                  = 0
eSILoadPreferringSchemalocation          = 1
eSILoadByNamespace                       = 2
eSICombiningBoth                         = 3
eSILicenseNamespaceOnly                  = 4
```

eSILoadBySchemalocation
Asigna a Schema Import el valor `LoadBySchemalocation`. El valor del atributo `schemaLocation` se usa para buscar el esquema, teniendo en cuenta las asignaciones de catálogo. Si está presente el atributo `namespace`, se importa el espacio de nombres.

 © 2014 Altova GmbH

eSILoadPreferringSchemalocation
Asigna a Schema Import el valor `LoadPreferringSchemalocation`. Si está presente el atributo `schemaLocation`, éste se utiliza teniendo en cuenta las asignaciones de catálogo. Si no está presente el atributo `schemaLocation`, se usa el valor del atributo `namespace` a través de una asignación de catálogo. Este literal es el **valor predeterminado** de la enumeración.

eSILoadByNamespace
Asigna a Schema Import el valor `LoadByNamespace`. El valor del atributo `namespace` se utiliza para encontrar el esquema a través de una asignación de catálogo.

eSICombiningBoth
Asigna a Schema Import el valor `CombiningBoth`. Si el atributo `namespace` o `schemaLocation` tiene una asignación de catálogo, esta asignación de catálogo se utiliza. Si ambos tienen una asignación de catálogo, entonces es el valor del parámetro `ENUMSchemaMapping` lo que decide cuál de las asignaciones se utiliza. Si ninguna tiene una asignación de catálogo, se usa el valor del atributo `schemaLocation` (que debería ser una URL).

eSILicenseNamespaceOnly
Asigna a Schema Import el valor `LicenseNamespaceOnly`. El espacio de nombres se importa. No se importa ningún documento de esquema.

ENUMSchemaMapping

Descripción
Contiene los literales de enumeración que definen cuál de las dos asignaciones de catálogo se prefiere: Esta enumeración sirve para eliminar ambigüedades en `ENUMLoadSchemalocation` y `ENUMSchemaImports`.

Utilizada por

Interfaz	*Operación*
`IXBRL`	`SchemaMapping`
`IXMLValidator`	`SchemaMapping`
`IXSLT`	`SchemaMapping`

Literales de la enumeración

```
eSMPreferSchemalocation          = 0
eSMPreferNamespace               = 1
```

eSMPreferSchemalocation
Establece que se debe seleccionar la URL de schema location.

eSMPreferNamespace
Establece que se debe seleccionar el espacio de nombres.

ENUMTableOutputFormat

Descripción
Contiene los literales de enumeración que definen el formato de salida del documento que contiene las tablas generadas.

Ustilizada por

Interfaz	*Operación*
IXBRL	TableOutputFormat

Literales de la enumeración

```
eFormatXML                  = 0
eFormatHTML                 = 1
```

eSMPreferSchemalocation
Establece la opción de asignación de esquemas para seleccionar la URL de la ubicación del esquema.

eSMPreferNamespace
Establece la opción de asignación de esquemas para seleccionar el espacio de nombres.

ENUMValidationType

Descripción
Contiene literales de enumeración que definen qué tipo de documento se debe validar.

Utilizada por

Interfaz	*Operación*
IXMLValidator	IsValid

 © 2014 Altova GmbH

Literales de la enumeración

```
eValidateAny                    = 0
eValidateXMLWithDTD             = 1
eValidateXMLWithXSD             = 2
eValidateDTD                    = 3
eValidateXSD                    = 4
```

eValidateAny
Establece que el tipo de validación es `Any`. Esto valida el documento después de detectar su tipo automáticamente.

eValidateXMLWithDTD
Establece que el tipo de validación es `XMLWithDTD`. Especifica que el documento XML debe validarse con un documento DTD.

eValidateXMLWithXSD
Establece que el tipo de validación es `XMLWithXSD`. Especifica que el documento XML debe validarse con un esquema XML.

eValidateDTD
Establece que el tipo de validación es `ValidateDTD`. Especifica que debe validarse un documento DTD.

eValidateXSD
Establece que el tipo de validación es `ValidateXSD`. Especifica que debe validarse un documento de XML Schema.

ENUMWellformedCheckType

Descripción
Contiene los literales de enumeración que definen el tipo de documento cuyo formato XML debe comprobarse: XML o DTD.

Utilizada por

Interfaz	*Operación*
IXMLValidator	IsWellFormed

Literales de la enumeración

```
eWellFormedAny          = 0
eWellFormedXML          = 1
eWellFormedDTD          = 2
```

eWellformedAny
Establece que el tipo de comprobación de formato es `Any`. Comprueba el formato XML del documento tras detectar automáticamente si se trata de un documento XML o DTD.

eWellformedXML
Establece que el tipo de comprobación de formato es `XML`. Comprueba si formato del documento XML se ajusta a la especificación XML 1.0 o XML 1.1.

eWellformedDTD
Establece que el tipo de comprobación de formato es `DTD`. Comprueba si formato del documento DTD es correcto.

ENUMXBRLValidationType

Descripción
Contiene literales de enumeración que definen el tipo de documento XBRL que se debe validar: una instancia XBRL o una taxonomía XBRL.

Utilizada por

Interfaz	*Operación*
IXBRL	IsValid

Literales de la enumeración

```
eValidateXBRLAny                    = 0
eValidateXBRLInstance               = 1
eValidateXBRLTaxonomy               = 2
```

eValidateXBRLAny
Establece que el tipo de validación es `Any`. Valida el documento tras detectar automáticamente de qué tipo se trata.

 © 2014 Altova GmbH

eValidateXBRLInstance
Establece que el tipo de validación es `Instance`. Valida los documentos XBRL de instancia.

eValidateXBRLTaxonomy
Establece que el tipo de validación es `Taxonomy`. Valida los documentos de taxonomía XBRL.

ENUMXMLValidationMode

Descripción
Contiene literales de enumeración que definen el modo de procesamiento XML que se debe usar: validación o comprobación de formato.

Utilizada por

Interfaz	*Operación*
IXMLValidator	XMLValidationMode
IXQuery	XMLValidationMode
IXSLT	XMLValidationMode

Literales de la enumeración

```
eXMLValidationModeWF          = 0
eXMLValidationModeID          = 1
eXMLValidationModeValid       = 2
```

eXMLValidationModeWF
Establece que el modo de procesamiento XML es `Wellformed`. Es el valor predeterminado.

eXMLValidationModeID
Solo para uso interno.

eXMLValidationModeValid
Establece que el modo de procesamiento XML es `Validation`.

ENUMXQueryVersion

Descripción

Contiene literales de enumeración que especifican qué versión de XQuery se debe usar: XQuery 1.0 o 3.0.

Utilizada por

Interfaz	*Operación*
IXQuery	EngineVersion

Literales de la enumeración

```
eXQVersion10        = 1
eXQVersion30        = 3
```

eXQVersion10

Establece que la versión XQuery es XQuery 1.0.

eXQVersion30

Establece que la versión XQuery es XQuery 3.0. Es el valor predeterminado.

ENUMXSDVersion

Descripción

Contiene literales de enumeración que indican qué versión de XML Schema se debe usar para la validación: XSD 1.0 o 1.1.

Utilizada por

Interfaz	*Operación*
IXMLValidator	XSDVersion
IXQuery	XSDVersion
IXSLT	XSDVersion

Literales de la enumeración

```
eXSDVersionAuto             = 0
eXSDVersion10               = 1
```

 © 2014 Altova GmbH

```
eXSDVersion11              = 2
```

eXSDVersionAuto
Establece que la versión de XML Schema que se debe usar es `Auto-detect`. La versión XSD se detecta automáticamente tras analizar el documento XSD. Si el atributo `vc:minVersion` del documento XSD tiene el valor `1.1`, se considera que el documento tiene la versión XSD 1.1. Si este atributo tiene cualquier otro valor o si el atributo no está en el documento, se entiende que el documento tiene la versión XSD 1.0.

eXSDVersion10
Establece que la versión de XML Schema que se debe usar es XML Schema 1.0.

eXSDVersion11
Establece que la versión de XML Schema que se debe usar es XML-Schema 1.1.

ENUMXSLTVersion

Descripción
Contiene literales de enumeración que especifican qué versión XSLT se debe usar: XSLT 1.0, 2.0 o 3.0.

Utilizada por

Interfaz	*Operación*
IXSLT	EngineVersion

Literales de la enumeración

```
eVersion10         = 1
eVersion20         = 2
eVersion30         = 3
```

eVersion10
Establece que la versión XSLT que se debe usar es XSLT 1.0.

eVersion20
Establece que la versión XSLT que se debe usar es XSLT 2.0.

eVersion30
Establece que la versión XSLT que se debe usar es XSLT 3.0.

© 2014 Altova GmbH

Altova RaptorXML+XBRL Server 2014

Información adicional

8 Información adicional

Esta sección contiene información adicional sobre:

- Parámetros de XBRL Formula

© 2014 Altova GmbH

8.1 Sugerencias sobre la ubicación del esquema

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 cómo trabajar con la información del atributo `schemaLocation` (*ver la descripción de la opción que aparece más arriba*). Recuerde que RaptorXML+XBRL 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.

8.2 Parámetros de XBRL Formula

Este apartado contiene varios temas:

- Formatos de los parámetros de fórmula, que ofrece ejemplos de los formatos XML y JSON de los parámetros de fórmulas XBRL.
- Usar parámetros de fórmulas contiene fragmentos de código Java, VB.NET, C#, VBScript y JScript que muestran cómo usar los parámetros de fórmulas con objetos de las bibliotecas Java y COM/.NET API.

© 2014 Altova GmbH

8.2.1 Formatos de los parámetros de fórmula

Los parámetros de fórmula pueden estar en formato XML o en formato JSON.

Formato XML
El ejemplo de código que aparece a continuación muestra parámetros de fórmula en formato XML.

```
<?xml version="1.0" encoding="utf-8"?>
<options:formula-parameters
  xmlns:options="http://www.altova.com/schemas/altova/raptorxml/options"
  xmlns:p="http://xbrl.org/formula/conformance/paramstuff"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.altova.com/schemas/altova/raptorxml/options
http://www.altova.com/schemas/altova/raptorxml/options.xsd">

  <options:parameter name="p1">
    <options:value type="xs:string">hello world from new xml (without namespace)
</options:value>
  </options:parameter>
  <options:parameter name="p:p1" type="xs:string" value="hello world from new
xml"/>

</options:formula-parameters>
```

Observe estos aspectos del ejemplo:

- El atributo `@type` es opcional y toma el valor predeterminado `xs:string`.
- Se pueden indicar varios elementos secundarios `<options:value>` para asignar una secuencia XPath a un parámetro.
- No se pueden usar `@value` y `<options:value>` al mismo tiempo.

Formato JSON
El ejemplo de código que aparece a continuación muestra parámetros de fórmula en formato JSON.

```
{
  "formula-parameters": [
  {
    "name": "p1",
    "values": [
      {
        "type": "xs:string",
        "value": "hello world from json new (without namespace)"
      }
    ]
   }, {
```

```
        "name": "ns1:p1",
        "values": [
          {
            "type": "xs:string",
            "value": "hello world from json new"
          }
         ]
       }
    ],
    "namespaces": {
      "xs": "http://www.w3.org/2001/XMLSchema",
      "ns1": "http://xbrl.org/formula/conformance/paramstuff"
    }
}
```

Observe estos aspectos del ejemplo:

- La clave `type` es opcional y toma el valor predeterminado `xs:string`.
- La clave `xs` es opcional y toma el valor predeterminado `http://www.w3.org/2001/XMLSchema`.
- El tipo de la asignación de parámetros se usa si un valor se especifica como cadena JSON directamente.
- También se puede usar :

```
{
    "name": "p2",
    "type": "xs:string",
    "value": "hello world from json new (without namespace)"
 }, {
    "name": "p3",
    "type": "xs:int",
    "values": ["1", "2"]
 }, {
    "name": "p4",
    "type": "xs: int",
    "values": ["1", {"type": "xs:string", "value": "abc"}, "2"]
 }
```

 © 2014 Altova GmbH

8.2.2 Usar parámetros de fórmula

Los fragmentos de código que aparecen más abajo muestran cómo usar parámetros de fórmulas XBRL en varios lenguajes de programación. Para Java, consulte el apartado clase XBRL de la API de Java. Para los demás lenguajes consulte el apartado interfaz XBRL de la API de COM/.NET.

Java

```
 RaptorXMLFactory rxml = RaptorXML.getFactory();
 XBRL xbrl = rxml.getXBRL();

 xbrl.addFormulaParameter( "ns1:string", "ns1:Param1", "ns1:theqname" );
 xbrl.addFormulaParameterNamespace( "ns1", "www.www.www" );

 // The parameter is an array of dates
 xbrl.addFormulaArrayParameter( "", "startDates", new Object[]{ new
FormulaParam( "xs:date", "2010-01-01" ), new FormulaParam( "xs:date",
"2012-01-01" ) } );

 // The parameter is an array of figs
 xbrl.addFormulaArrayParameter( "ns1:figs", "startFigs", new Object[]
{ "fig1", "fig2", "fig3" } );

 // The parameter is an array of figs, dates and raisins (rather wild
example)
 xbrl.addFormulaArrayParameter( "ns1:figs", "startDryFruit", new Object[]
{ "fig1", "fig2", new FormulaParam( "xs:date", "2010-01-01" ), new
FormulaParam( "ns1:raisin", "dried grape" ), "fig3" } );
```

VB.NET

```
Dim objRaptor As New Server()
   Dim objXBRL As XBRL
   objXBRL = objRaptor.GetXBRL()

   objXBRL.AddFormulaParameter("ns1:string", "ns1:Param1", "ns1:theqname")
   objXBRL.AddFormulaParameterNamespace("ns1", "www.www.www")

   'The parameter is an array of dates
   objXBRL.AddFormulaArrayParameter("", "startDates", {New XBRLFormulaParam
 With {.ParamType = "xs:date", .ParamValue = "2010-01-01"}, New
XBRLFormulaParam With {.ParamType = "xs:date", .ParamValue = "2012-01-
01"}})

   'The parameter is an array of figs
   objXBRL.AddFormulaArrayParameter("ns1:figs", "startFigs", {"fig1",
"fig2", "fig3"})

   'The parameter is an array of figs, dates and raisins (rather wild
example)
   objXBRL.AddFormulaArrayParameter("ns1:figs", "startDryFruit", {"fig1",
```

```
"fig2", New XBRLFormulaParam With {.ParamType = "xs:date", .ParamValue =
"2010-01-01"}, New XBRLFormulaParam With {.ParamType =
"ns1:raisin", .ParamValue = "dried grape"}, "fig3"})
```

C#

```
Server app = new Server();
   XBRL objXBRL = app.GetXBRL();

   objXBRL.AddFormulaParameter("ns1:string", "ns1:Param1", "ns1:theqname");
   objXBRL.AddFormulaParameterNamespace("ns1", "www.www.www");

   //The parameter is an array of dates
   objXBRL.AddFormulaArrayParameter("", "startDates", new object[] {new
XBRLFormulaParam { ParamType = "xs:date", ParamValue = "2010-01-01"}, new
XBRLFormulaParam {ParamType = "xs:date", ParamValue = "2012-01-01"}});

   //The parameter is an array of figs
   objXBRL.AddFormulaArrayParameter("ns1:figs", "startFigs", new object[]
{"fig1", "fig2", "fig3"});

   //The parameter is an array of figs, dates and raisins (rather wild
example)
   objXBRL.AddFormulaArrayParameter("ns1:figs", "startDryFruit", new
object[] { "fig1", "fig2", new XBRLFormulaParam { ParamType = "xs:date",
ParamValue = "2010-01-01" }, new XBRLFormulaParam { ParamType =
"ns1:raisin", ParamValue = "dried grape" }, "fig3" });
```

VBScript

Como la biblioteca de tipos de RaptorXML no se puede cargar con lenguajes de scripting y el tipo `XBRLFormulaParameters` no existe, el usuario de VBScript debe declarar una clase en su programa, en lugar de usar objetos `XBRL.FormulaParam`. La clase debe tener dos propiedades públicas: `ParamName` y `ParamValue` (al igual que `XBRL.FormulaParam`). La clase debería tener un constructor que tome el tipo y el valor (porque esto facilita su uso). De lo contrario, es necesario crear el objeto y definir sus miembros. Para más información consulte el apartado interfaz XBRL de la API de COM/.NET.

```
Class MyPair
   Public ParamType
   Public ParamValue
   Public Default Function Init( inType, inValue )
          ParamType = inType
          ParamValue = inValue
          set Init = Me
   End Function
End Class
```

 © 2014 Altova GmbH

```
Sub Main
   Dim objRaptor
   Set objRaptor = WScript.GetObject( "", "RaptorXML.Server" )
   Dim objXBRL
   Set objXBRL = objRaptor.GetXBRL

   Call objXBRL.AddFormulaParameter("ns1:string", "ns1:Param1",
"ns1:theqname")
   Call objXBRL.AddFormulaParameterNamespace("ns1", "www.www.www")

   'The parameter is an array of dates
   Call objXBRL.AddFormulaArrayParameter("", "startDates", Array( ( New
MyPair)( "xs:date", "2010-01-01"), ( New MyPair)( "xs:date", "2012-01-
01") ) )

   'The parameter is an array of figs
   Call objXBRL.AddFormulaArrayParameter("ns1:figs", "startFigs",
Array("fig1", "fig2", "fig3") )

   'The parameter is an array of figs, dates and raisins (rather wild
example)
   Call objXBRL.AddFormulaArrayParameter("ns1:figs", "startDryFruit",
Array("fig1", "fig2", ( New MyPair)("xs:date", "2010-01-01"(, ( New MyPair)
("ns1:raisin", "dried grape"), "fig3") )
End Sub

Call Main
```

JScript

Como la biblioteca de tipos de RaptorXML no se puede cargar con lenguajes de scripting y el tipo `XBRLFormulaParameters` no existe, el usuario de JScript debe declarar clases función en su programa que almacene el par valor/tipo, en lugar de usar objetos `XBRL.FormulaParam`. El nombre de los miembros debe ser `ParamType` y `ParamValue`. Para más información consulte el apartado interfaz XBRL de la API de COM/.NET.

```
function FormulaParam( inType, inValue)
{
   this.ParamType = inType;
   this.ParamValue = inValue;
}

function main()
{
   var objRaptor = new ActiveXObject( "RaptorXML.Server" );
   var objXBRL = objRaptor.GetXBRL();

   objXBRL.addFormulaParameter( "ns1:string", "ns1:Param1",
"ns1:theqname" );
```

```
   objXBRL.addFormulaParameter( "xs:string", "Param1", "bla",
"www.www.www" );

   // The parameter is an array of dates
   objXBRL.addFormulaArrayParameter("", "startDates", [new
FormulaParam("xs:date", "2010-01-01"), new FormulaParam("xs:date", "2012-
01-01")]);

   // The parameter is an array of figs
   objXBRL.addFormulaArrayParameter("ns1:figs", "startFigs", ["fig1",
"fig2", "fig3"]);

   // The parameter is an array of figs, dates and raisins (rather wild
example)
   objXBRL.addFormulaArrayParameter("ns1:figs", "startDryFruit", ["fig1",
"fig2", new FormulaParam("xs:date", "2010-01-01"), new
FormulaParam("ns1:raisin", "dried grape"), "fig3"]);
}

main()
```

 © 2014 Altova GmbH

Altova RaptorXML+XBRL Server 2014

Información sobre motores XSLT y XQuery

9 Información sobre motores XSLT y XQuery

Los motores XSLT y XQuery de RaptorXML+XBRL Server siguen las especificaciones del W3C y, por tanto, son más estrictos que otros motores anteriores de Altova, como los de las versiones antiguas de XMLSpy y del predecesor de RaptorXML+XBRL Server, el procesador descatalogado AltovaXML. Por consiguiente, RaptorXML+XBRL Server señala algunos errores leves que antes no se notificaban en la versión anterior de estos motores.

Por ejemplo:

- Se notifica un error de tipo (`err:XPTY0018`) si el resultado de un operador de ruta de acceso contiene tanto nodos como no nodos.
- Se notifica un error de tipo (`err:XPTY0019`) si `E1` en una expresión XPath `E1/E2` no da como resultado una secuencia de nodos.

Si encuentra este tipo de errores, modifique el documento XSLT/XQuery o el documento de instancia según corresponda.

Esta sección describe características relacionadas con la implementación de los motores e incluye estos apartados:

- XSLT 1.0
- XSLT 2.0
- XSLT 3.0
- XQuery 1.0
- XQuery 3.0

 © 2014 Altova GmbH

9.1 XSLT 1.0

El motor XSLT 1.0 de RaptorXML+XBRL Server cumple con la recomendación XSLT 1.0 del 16 de noviembre de 1999 y con la recomendación XPath 1.0 del 16 de noviembre de 1999, ambas del W3C.

Nota sobre la implementación
Cuando el atributo `method` de `xsl:output` tiene el valor HTML o si selecciona de forma predeterminada el formato de salida HTML, los caracteres especiales del archivo XML o XSLT se insertan en el documento HTML como referencias de caracteres HTML. Por ejemplo, el carácter ` ` (la referencia de carácter decimal para un espacio de no separación) se inserta como ` ` en el código HTML.

9.2 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+XBRL Server cumple con la recomendación XSLT 2.0 del 23 de enero de 2007 y la recomendación XPath 2.0 del 14 de diciembre de 2010, ambas del W3C.

Compatibilidad con versiones antiguas

El motor XSLT 2.0 es compatible con versiones previas. Esto 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="http://www.w3.org/2005/xpath-functions"
    ...
</xsl:stylesheet>
```

 © 2014 Altova GmbH

Es necesario tener en cuenta que:

- El motor XSLT 2.0 utiliza el espacio de nombres Funciones XPath 2.0 y XQuery 1.0 como **espacio de nombres de funciones predeterminado**. Esto significa que puede usar funciones XPath 2.0 y XSLT 2.0 en su hoja de estilos sin prefijos. Si declara el espacio de nombres Funciones XPath 2.0 en su hoja de estilos con un prefijo, podrá usar el prefijo asignado en la declaración.
- Cuando se usan constructores de tipo y tipos del espacio de nombres XML Schema, el prefijo utilizado en la declaración de espacio de nombres se debe usar en la llamada al constructor de tipo (por ejemplo, `xs:date`).
- Algunas funciones XPath 2.0 se llaman igual que algunos tipos de datos de XML Schema. Por ejemplo, las funciones XPath `fn:string` y `fn:boolean` y los tipos de datos de XML Schema `xs:string` y `xs:boolean`. Por tanto, si usa la expresión `string('Hello')`, la expresión se evalúa como `fn:string('Hello')` y no como `xs:string('Hello')`.

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`.

© 2014 Altova GmbH

9.3 XSLT 3.0

El motor XSLT 3.0 de RaptorXML+XBRL Server cumple con el borrador de trabajo XSLT 3.0 Last Call del 12 de diciembre de 2013 y con la recomendación XPath 3.0 del 8 de abril de 2014, ambos del W3C.

El motor XSLT 3.0 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.

Nota: estas instrucciones XSLT 3.0 no son compatibles por ahora:

```
xsl:accept
xsl:accumulator
xsl:accumulator-rule
xsl:assert
xsl:break
xsl:context-item
xsl:expose
xsl:fork
xsl:iterate
xsl:map
xsl:map-entry
xsl:merge
xsl:merge-action
xsl:merge-key
xsl:merge-source
xsl:mode
xsl:next-iteration
xsl:next-match
xsl:on-completion
xsl:override
xsl:package
xsl:stream
xsl:use-package
```

9.4 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

Especificaciones compatibles
El motor XQuery 1.0 de RaptorXML+XBRL Server cumple con la recomendación XQuery 1.0 del 14 de diciembre de 2010 del W3C. El estándar XQuery concede libertad a la hora de implementar muchas características. A continuación explicamos cómo se implementaron estas características en el motor XQuery 1.0 de RaptorXML+XBRL Server.

Compatibilidad con esquemas
El motor XQuery 1.0 está preparado para esquemas.

Codificación
El motor XQuery 1.0 es compatible con las codificaciones de caracteres UTF-8 y UTF-16.

Espacios de nombres
Se predefinen estos URI de espacios de nombres y sus enlaces asociados.

Espacio de nombres	Prefijo	URI del espacio de nombres
Tipos XML Schema	`xs:`	`http://www.w3.org/2001/XMLSchema`
Schema instance	`xsi:`	`http://www.w3.org/2001/XMLSchema-instance`
Funciones	`fn:`	`http://www.w3.org/2005/xpath-functions`

 © 2014 Altova GmbH

integradas		
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.

 © 2014 Altova GmbH

Precisión de tipos numéricos

- El tipo de datos `xs:integer` es de precisión arbitraria, es decir, puede representar un número de dígitos cualquiera.
- El tipo de datos `xs:decimal` tiene un límite de 20 dígitos después del punto decimal.
- Los tipos de datos `xs:float` y `xs:double` tienen una precisión limitada de 15 dígitos.

Compatibilidad con instrucciones XQuery

La instrucción `Pragma` no es compatible. Si se encuentra, se ignora y en su lugar se evalúa la expresión de reserva.

9.5 XQuery 3.0

El motor XQuery 3.0 de RaptorXML+XBRL Server cumple con la recomendación XQuery 3.0 del 8 de abril de 2014 del W3C y es compatible con funciones XPath y XQuery 3.0

Tiene las mismas características de implementación que el motor XQuery 1.0.

Nota: estas características no son compatibles por ahora:

Cláusula Group By en expresiones FLWOR (3.10.7 Cláusula Group By).
Cláusulas Tumbling Window y Sliding Window en expresiones FLWOR (3.10.4 Cláusula Window).
Cláusula Count en expresiones FLWOR (3.10.6 Cláusula Count).
Allowing empty en la cláusula 3.10.2 For, para funciones similares a las combinaciones externas de SQL.
Expresiones Try/Catch (3.15 Expresiones Try/Catch).
Constructores de espacios de nombres calculados (3.9.3.7 Computed Namespace Constructors).
Declaraciones de salida (2.2.4 Serialización).
Anotaciones (4.15 Anotaciones).
Aserciones de función en pruebas de función.

 © 2014 Altova GmbH

Altova RaptorXML+XBRL Server 2014

Funciones XSTL y XPath/XQuery

10 Funciones XSTL y XPath/XQuery

Esta sección enumera las funciones de extensión de Altova y otras funciones de extensión que se pueden utilizar con expresiones XPath y XQuery. Las funciones de extensión de Altova se pueden usar con los motores XSLT y XQuery de Altova y ofrecen algunas funciones más aparte de las que están disponibles en las bibliotecas de funciones definidas en los estándares del W3C.

Aspectos generales

Es necesario tener en cuenta estos puntos generales:

- A las funciones de las bibliotecas de funciones `core` definidas en las especificaciones W3C se les puede llamar sin un prefijo. Esto se debe a que los motores XSLT y XQuery leen funciones sin prefijo como si pertenecieran a un espacio de nombres de funciones predeterminado, que es el que se especifica en las especificaciones de las funciones XPath y XQuery`http://www.w3.org/2005/xpath-functions`. Si este espacio de nombres se declara explícitamente en un documento XSLT o XQuery, el prefijo utilizado en la declaración de espacio de nombres también se puede usar en el nombre de las funciones.
- Por lo general, si una función espera como argumento una secuencia de un elemento y se suministra una secuencia de más de un elemento, entonces se devuelve un error.
- Todas las comparaciones de cadena se realizan 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 deben comparar. Cuando el uso horario no se conoce de forma explícita, se usa el uso horario implícito, que se toma del reloj del sistema. Para probar cuál es su valor puede utilizar 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 (*ver más abajo*). Las comparaciones de cadenas de texto (incluidas las que usan 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 predeterminada de puntos de código Unicode.

 © 2014 Altova GmbH

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`: jaonés	`ja_JP`
`nb`: noruego bokmål	`nb_NO`
`nl`: holandés	`nl_AW, nl_BE, nl_NL`
`nn`: noruego nynorsk	`nn_NO`
`pt`: portugués	`pt_AO, pt_BR, pt_GW, pt_MZ, pt_PT, pt_ST`
`ru`: ruso	`ru_MD, ru_RU, ru_UA`
`sv`: sueco	`sv_FI, sv_SE`

Eje del espacio de nombres

El eje del espacio de nombres se dejó de utilizar en XPath 2.0. Sin embargo, los ejes de espacio de nombres son compatibles con la aplicación. 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()`.

10.1 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. Tenga en cuenta que en futuras versiones del producto algunas funciones pueden dejar de ser compatibles o su comportamiento puede cambiar. Por tanto, consulte siempre la documentación del producto para conocer el funcionamiento de estas funciones en cada versión del producto.

Las funciones definidas en las especificaciones XPath/XQuery Functions del W3C se pueden usar en (i) expresiones XPath en contextos XSLT y en (ii) expresiones XQuery en documentos XQuery. En esta documentación las funciones que se pueden usar en el primer contexto (XPath en XSLT) llevan el símbolo **XP** y se les llama funciones XPath. Las funciones que se pueden usar en contextos XQuery llevan el símbolo **XQ** y funcionan como funciones XQuery. Las especificaciones XSLT del W3C también definen funciones que se pueden usar en expresiones XPath en documentos XSLT. Estas funciones llevan el símbolo **XSLT** y se les denomina funciones XSLT. Por cada función se indica en qué versión de XPath/XQuery y XSLT se puede usar (ver símbolos más abajo). Las funciones de las bibliotecas de funciones XPath/XQuery y XSLT aparecen sin prefijo. Las funciones de extensión de otras bibliotecas, como las funciones de extensión de Altova, aparecen con un prefijo.

Funciones XPath (en expresiones XPath en XSLT):	**XP1** **XP2** **XP3**
Funciones XSLT (en expresiones XPath en XSLT):	**XSLT1** **XSLT2** **XSLT3**
Funciones XQuery (en expresiones XQuery en XQuery):	**XQ1** **XQ3**

Funciones XSLT
Estas funciones solo se pueden usar en expresiones XPath en contextos XSLT (igual que las funciones XSLT 2.0 `current-group()` o `key()`). Estas funciones no funcionan en otro contexto que no sea XSLT (por ejemplo, en contextos XQuery). Recuerde que las funciones XSLT para XBRL solamente se pueden usar en las ediciones de los productos de Altova que son compatibles con XBRL.

Funciones XPath/XQuery
Las funciones XPath/XQuery (generales, de fecha y hora y de cadena) se pueden usar tanto en expresiones XPath en contextos XSLT como en expresiones XQuery.

Funciones para gráficos (solo en las ediciones Enterprise y Server Edition)
Las funciones de extensión para gráficos de Altova son compatibles con las ediciones Enterprise y Server Edition de los productos de Altova solamente. Estas funciones permiten generar gráficos a partir de datos XML.

Funciones para códigos de barras

 © 2014 Altova GmbH

Las funciones de extensión para códigos de barras de Altova permiten generar códigos de barras y colocarlos en los resultados generados con hojas de estilos XSLT.

10.1.1 Funciones XSLT

Las **funciones de extensión XSLT** pueden utilizarse en expresiones XPath en contextos XSLT y no funcionan en contextos que no sean XSLT (por ejemplo, en contextos XQuery).

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. Tenga en cuenta que en futuras versiones del producto algunas funciones pueden dejar de ser compatibles o su comportamiento puede cambiar. Por tanto, consulte siempre la documentación del producto para conocer el funcionamiento de estas funciones en cada versión del producto.

Funciones XPath (en expresiones XPath en XSLT):	XP1 XP2 XP3
Funciones XSLT (en expresiones XPath en XSLT):	XSLT1 XSLT2 XSLT3
Funciones XQuery (en expresiones XQuery en XQuery):	XQ1 XQ3

---

distinct-nodes [altova:]

altova:distinct-nodes(*node()****)** como **node()*** XSLT1 XSLT2 XSLT3

Toma un conjunto de nodos como entrada y devuelve el mismo conjunto menos los nodos que tengan el mismo valor (es decir, devuelve los nodos que son únicos). La comparación se hace con la función XPath/XQuery `fn:deep-equal`.

Ejemplo

- **altova:distinct-nodes**(`country`) devuelve todos los nodos secundarios `country` excepto los que tengan el mismo valor.

evaluate [altova:]

altova:evaluate(ExpresiónXPath *as xs:string***[, ValorDe$p1, ... ValorDe$pN])** XSLT1 XSLT2 XSLT3

Toma una expresión XPath, pasada como cadena, como argumento obligatorio. Devuelve el resultado de la expresión evaluada. Por ejemplo, **altova:evaluate('//Name[1]')** devuelve el contenido del primer elemento `Name` del documento. Observe que para pasar la expresión `//Name[1]` como cadena basta con ponerla entre comillas simples.

La función `altova:evaluate` puede tomar más argumentos, que son los valores de las variables del ámbito que se llaman `p1`, `p2`, `p3`... `pN`. Recuerde que (i) las variables deben definirse con nombres de tipo `pX`, siendo `X` un entero; (ii) los argumentos de la función `altova:evaluate` (*ver firma más abajo*), a partir del segundo argumento, ofrecen los valores de las variables, correspondiendo la secuencia de argumentos a la secuencia numérica de variables: `p1` corresponde a `pN` y el segundo argumento será el valor de la variable `p1`, el tercer argumento al de la variable `p2`, y así sucesivamente; (iii) los valores de las variables deben ser de tipo `item*`.

Ejemplo

 © 2014 Altova GmbH

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

En el ejemplo anterior puede observar que:

- El segundo argumento de la expresión `altova:evaluate` es el valor asignado a la variable `$p1`, el tercer argumento es el valor asignado a la variable `$p2` y así sucesivamente.
- Observe que el cuarto argumento de la función es un valor de cadena porque va entre comillas simples.
- El atributo `select` del elemento `xs:variable` suministra la expresión XPath. Como esta expresión debe ser de tipo `xs:string`, se pone entre comillas simples.

Más ejemplos

- ```
  <xsl:variable name="xpath" select="'$p1'" />
  <xsl:value-of select="altova:evaluate($xpath, //Name[1])" />
  ```
  El resultado es el valor del primer elemento `Name`.

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

La función de extensión `altova:evaluate()` es muy práctica cuando una expresión XPath de la hoja de estilos XSLT contiene partes que se deben evaluar de forma dinámica. Por ejemplo, imagine que el usuario selecciona un criterio de ordenación y este criterio se almacena en el atributo `UserReq/@sortkey`. En la hoja de estilos podría tener esta expresión:

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

La función `altova:evaluate()` lee el atributo `sortkey` del elemento secundario `UserReq` del primario del nodo de contexto. Imagine que el valor del atributo `sortkey` es `Price`. En ese caso, la función `altova:evaluate()` devuelve `Price`, que se convierte en el valor del atributo `select`:

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

Si esta instrucción `sort` aparece dentro del contexto de un elemento llamado `Order`, entonces los elementos `Order` se ordenan según el valor de los secundarios `Price`. Otra opción es que, si el valor de `@sortkey` fuera `Date`, por ejemplo, entonces los elementos `Order` se ordenarían según el valor de los secundarios `Date`. Es decir, el criterio de ordenación para `Order` se selecciona del atributo `sortkey` en tiempo de ejecución. Esto no sería posible con una expresión como:

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

En este caso, el criterio de ordenación sería el propio atributo `sortkey`, no `Price` ni `Date` (ni otro contenido actual de `sortkey`).

Nota: el contexto estático incluye espacios de nombres, tipos y funciones (pero no variables) del entorno de llamada. El URI base y el espacio de nombres predeterminado se heredan.

Más ejemplos

- Variables estáticas: `<xsl:value-of select="$i3, $i2, $i1" />`
  El resultado es los valores de las tres variables.
- Expresión XPath dinámica con variables dinámicas:
  ```
  <xsl:variable name="xpath" select="'$p3, $p2, $p1'" />
  <xsl:value-of select="altova:evaluate($xpath, 10, 20, 30)" />
  ```
  El resultado es "30 20 10"
- Expresión XPath dinámica sin variables dinámicas:
  ```
  <xsl:variable name="xpath" select="'$p3, $p2, $p1'" />
  <xsl:value-of select="altova:evaluate($xpath)" />
  ```
  Error: no se definió la variable para $p3.

encode-for-rtf [altova:]

altova:encode-for-rtf(input *as xs:string*, **preserveallwhitespace** *as xs:boolean*, **preservenewlines** *as xs:boolean*) como **xs:string** XSLT2 XSLT3

Convierte la cadena de entrada en código para RTF. Los espacios en blanco y las líneas nuevas se conservan o no dependiendo del valor booleano especificado para los correspondientes parámetros.

[Subir]

Funciones XBRL

Las funciones XBRL de Altova solo funcionan en las ediciones de los productos de Altova que son compatibles con XBRL.

xbrl-footnotes [altova:]

altova:xbrl-footnotes(*node()***)** como **node()*** XSLT2 XSLT3

Toma un nodo como argumento de entrada y devuelve el conjunto de nodos de nota al pie XBRL al que hace referencia el nodo de entrada.

xbrl-labels [altova:]

altova:xbrl-labels(*xs:QName*, *xs:string***)** como **node()*** XSLT2 XSLT3

Toma dos argumentos de entrada: un nombre de nodo y la ubicación del archivo de taxonomía en el que está el nodo. La función devuelve los nodos de etiqueta XBRL asociados al nodo de entrada.

[Subir]

 © 2014 Altova GmbH

10.1.2 Funciones XPath/XQuery: fecha y hora

Las funciones de extensión de fecha y hora de Altova se pueden usar en expresiones XPath y XQuery y permiten procesar datos almacenados en tipos de datos XML Schema de fecha y hora. Estas funciones se pueden usar con los **motores XPath 3.0 y XQuery 3.0** de Altova.

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. Tenga en cuenta que en futuras versiones del producto algunas funciones pueden dejar de ser compatibles o su comportamiento puede cambiar. Por tanto, consulte siempre la documentación del producto para conocer el funcionamiento de estas funciones en cada versión del producto.

Funciones XPath (en expresiones XPath en XSLT):	XP1 XP2 XP3
Funciones XSLT (en expresiones XPath en XSLT):	XSLT1 XSLT2 XSLT3
Funciones XQuery (en expresiones XQuery en XQuery):	XQ1 XQ3

Estas son las funciones de fecha y hora disponibles en contextos XPath/XQuery, agrupados según su funcionalidad:

- Agregan una duración (años, meses, días, horas, minutos o segundos) a xs:dateTime y devuelven xs:dateTime
- Agregan una duración (años, meses o días) a xs:date y devuelven xs:date
- Agregan una duración (horas, minutos o segundos) a xs:time y devuelven xs:time
- Quitan la zona horaria de funciones que generan la fecha y hora actual
- Devuelven un día de la semana de una fecha como número entero
- Generan la fecha, hora y duración a partir de los componentes léxicos de cada tipo
- Funciones para calcular la edad

Funciones de fecha y hora por orden alfabético

altova:add-days-to-date
altova:add-days-to-dateTime
altova:add-hours-to-dateTime
altova:add-hours-to-time
altova:add-minutes-to-dateTime
altova:add-minutes-to-time
altova:add-months-to-date
altova:add-months-to-dateTime
altova:add-seconds-to-dateTime
altova:add-seconds-to-time
altova:add-years-to-date
altova:add-years-to-dateTime
altova:age
altova:age-details
altova:build-date
altova:build-duration

altova:build-time
altova:current-dateTime-no-TZ
altova:current-date-no-TZ
altova:current-time-no-TZ
altova:weekday-from-date
altova:weekday-from-dateTime

[Subir]

Agregar una duración a xs:dateTime XP3 XQ3

Estas funciones sirven para agregar una duración a **xs:dateTime** y devuelven **xs:dateTime**. El tipo xs:dateTime tiene el formato CCYY-MM-DDThh:mm:ss.sss. Se trata de la concatenación de los formatos xs:date y xs:time separados por la letra T. Si quiere, puede usar un sufijo de zona horaria+01:00 (por ejemplo).

add-years-to-dateTime [altova:]

altova:add-years-to-dateTime(FechaHora *as xs:dateTime*, **Años** *as xs:integer***)** como **xs:dateTime** XP3 XQ3

Añade una duración en años un valor de fecha y hora. El segundo argumento es el número de años que se debe añadir al valor de fecha y hora dado como primer argumento. El resultado es de tipo xs:dateTime.

Ejemplos

- **altova:add-years-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), 10) devuelve 2024-01-15T14:00:00
- **altova:add-years-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), -4) devuelve 2010-01-15T14:00:00

add-months-to-dateTime [altova:]

altova:add-months-to-dateTime(FechaHora *as xs:dateTime*, **Meses** *as xs:integer***)** como **xs:dateTime** XP3 XQ3

Añade una duración en meses a un valor de fecha y hora. El segundo argumento es el número de meses que se debe añadir al valor de fecha y hora dado como primer argumento. El resultado es de tipo xs:dateTime.

Ejemplos

- **altova:add-months-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), 10) devuelve 2014-11-15T14:00:00
- **altova:add-months-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), -2) devuelve 2013-11-15T14:00:00

add-days-to-dateTime [altova:]

altova:add-days-to-dateTime(FechaHora *as xs:dateTime*, **Días** *as xs:integer***)** como **xs:dateTime** XP3 XQ3

Añade una duración en días a un valor de fecha y hora. El segundo argumento es el número de días que se deben añadir al valor de fecha y hora dado como primer argumento. El resultado es de tipo xs:dateTime.

Ejemplos

 © 2014 Altova GmbH

- **altova:add-days-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), 10) devuelve 2014-01-25T14:00:00
- **altova:add-days-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), -8) devuelve 2014-01-07T14:00:00

add-hours-to-dateTime [altova:]

altova:add-hours-to-dateTime(FechaHora *as xs:dateTime*, **Horas** *as xs:integer***)** como **xs:dateTime** **XP3** **XQ3**

Añade una duración en horas a un valor de fecha y hora. El segundo argumento es el número de horas que se deben añadir al valor de fecha y hora dado como primer argumento. El resultado es de tipo xs:dateTime.

Ejemplos

- **altova:add-hours-to-dateTime**(xs:dateTime("2014-01-15T13:00:00"), 10) devuelve 2014-01-15T23:00:00
- **altova:add-hours-to-dateTime**(xs:dateTime("2014-01-15T13:00:00"), -8) devuelve 2014-01-15T05:00:00

add-minutes-to-dateTime [altova:]

altova:add-minutes-to-dateTime(FechaHora *as xs:dateTime*, **Minutos** *as xs:integer***)** como **xs:dateTime** **XP3** **XQ3**

Añade una duración en minutos a un valor de fecha y hora. El segundo argumento es el número de minutos que se debe añadir al valor de fecha y hora dado como primer argumento. El resultado es de tipo xs:dateTime.

Ejemplos

- **altova:add-minutes-to-dateTime**(xs:dateTime("2014-01-15T14:10:00"), 45) devuelve 2014-01-15T14:55:00
- **altova:add-minutes-to-dateTime**(xs:dateTime("2014-01-15T14:10:00"), -5) devuelve 2014-01-15T14:05:00

add-seconds-to-dateTime [altova:]

altova:add-seconds-to-dateTime(FechaHora *as xs:dateTime*, **Segundos** *as xs:integer***)** como **xs:dateTime** **XP3** **XQ3**

Añade una duración en segundos a un valor de fecha y hora. El segundo argumento es el número de segundos que se debe añadir al valor de fecha y hora dado como primer argumento. El resultado es de tipo xs:dateTime.

Ejemplos

- **altova:add-seconds-to-dateTime**(xs:dateTime("2014-01-15T14:00:10"), 20) devuelve 2014-01-15T14:00:30
- **altova:add-seconds-to-dateTime**(xs:dateTime("2014-01-15T14:00:10"), -5) devuelve 2014-01-15T14:00:05

[Subir]

Agregar una duración a `xs:date` XP3 XQ3

Estas funciones agregan una duración a **`xs:date`** y devuelven **`xs:date`**. El tipo `xs:date` tiene el formato `CCYY-MM-DD`.

add-years-to-date [altova:]

`altova:add-years-to-date(Fecha` *`as xs:date`*`, Años` *`as xs:integer`*`)` como **`xs:date`**
XP3 XQ3

Añade una duración en años a una fecha. El segundo parámetro es el número de años que se debe añadir a la fecha dada como primer argumento. El resultado es de tipo `xs:date`.

Ejemplos

- **`altova:add-years-to-date`**`(xs:date("2014-01-15"), 10)` devuelve `2024-01-15`
- **`altova:add-years-to-date`**`(xs:date("2014-01-15"), -4)` devuelve `2010-01-15`

add-months-to-date [altova:]

`altova:add-months-to-date(Fecha` *`as xs:date`*`, Meses` *`as xs:integer`*`)` como **`xs:date`**
XP3 XQ3

Añade una duración en meses a una fecha. El segundo argumento es el número de meses que se debe añadir a la fecha dada como primer argumento. El resultado es de tipo `xs:date`.

Ejemplos

- **`altova:add-months-to-date`**`(xs:date("2014-01-15"), 10)` devuelve `2014-11-15`
- **`altova:add-months-to-date`**`(xs:date("2014-01-15"), -2)` devuelve `2013-11-15`

add-days-to-date [altova:]

`altova:add-days-to-date(Fecha` *`as xs:date`*`, Días` *`as xs:integer`*`)` como **`xs:date`**
XP3 XQ3

Añade una duración en días a una fecha. El segundo argumento es el número de días que se deben añadir a la fecha dad como primer argumento. El resultado es de tipo `xs:date`.

Ejemplos

- **`altova:add-days-to-date`**`(xs:date("2014-01-15"), 10)` devuelve `2014-01-25`
- **`altova:add-days-to-date`**`(xs:date("2014-01-15"), -8)` devuelve `2014-01-07`

[Subir]

Agregar una duración a `xs:time` XP3 XQ3

Estas funciones agregan una duración a **`xs:time`** y devuelven **`xs:time`**. El tipo `xs:time` tiene un formato léxico de este tipo `hh:mm:ss.sss`. Si quiere, puede añadir un sufijo de zona horaria. La letra `Z` indica (UTC). Las demás zonas horarias se representan con la diferencia que hay entre ellas y la zona UTC: `+hh:mm` o `-hh:mm`. Si falta el valor de zona horaria, se entiende que se desconoce (no se da por hecho que es UTC)

add-hours-to-time [altova:]

`altova:add-hours-to-time(Hora` *`as xs:time`*`, Horas` *`as xs:integer`*`)` como **`xs:time`**
XP3 XQ3

Añade una duración en horas a una hora. El segundo argumento es el número de horas que

se debe añadir a la hora dada como primer argumento. El resultado es de tipo `xs:time`.

Ejemplos

- **altova:add-hours-to-time**(xs:time("11:00:00"), 10) devuelve 21:00:00
- **altova:add-hours-to-time**(xs:time("11:00:00"), -7) devuelve 04:00:00

add-minutes-to-time [altova:]

altova:add-minutes-to-time(Hora *as xs:time*, **Minutos** *as xs:integer*) como **xs:time** XP3 XQ3

Añade una duración en minutos a una hora. El segundo argumento es el número de minutos que se debe añadir a la hora dada como primer argumento. El resultado es de tipo `xs:time`.

Ejemplos

- **altova:add-minutes-to-time**(xs:time("14:10:00"), 45) devuelve 14:55:00
- **altova:add-minutes-to-time**(xs:time("14:10:00"), -5) devuelve 14:05:00

add-seconds-to-time [altova:]

altova:add-seconds-to-time(Hora *as xs:time*, **Segundos** *as xs:integer*) como **xs:time** XP3 XQ3

Añade una duración en segundos a una hora. El segundo argumento es el número de segundos que se debe añadir a la hora dada como primer argumento. El resultado es de tipo `xs:time`. El componente **Segundos** puede estar comprendido entre 0 y 59.999.

Ejemplos

- **altova:add-seconds-to-time**(xs:time("14:00:00"), 20) devuelve 14:00:20
- **altova:add-seconds-to-time**(xs:time("14:00:00"), 20.895) devuelve 14:00:20.895

[Subir]

Quitar la parte de zona horaria de los tipos de datos date/time XP3 XQ3

Estas funciones quitan la zona horaria de los valores **xs:dateTime**, **xs:date** o **xs:time** actuales. Tenga en cuenta que la diferencia entre `xs:dateTime` y `xs:dateTimeStamp` es que en esta última la parte de zona horaria es obligatoria (mientras que en la primera es opcional). Es decir, el formato de un valor `xs:dateTimeStamp` puede ser `CCYY-MM-DDThh:mm:ss.sss±hh:mm` o `CCYY-MM-DDThh:mm:ss.sssZ`. Si la fecha y la hora se leen del reloj del sistema como `xs:dateTimeStamp`, la función `current-dateTime-no-TZ()` se puede usar para quitar la zona horaria.

current-dateTime-no-TZ [altova:]

altova:current-dateTime-no-TZ() como **xs:dateTime** XP3 XQ3

Esta función no toma ningún argumento. Quita la parte de zona horaria de `current-dateTime()` (que es la fecha y hora actual según el reloj del sistema) y devuelve un valor de tipo `xs:dateTime`.

Ejemplos

Si la fecha y hora actual es **2014-01-15T14:00:00+01:00**:

- **altova:current-dateTime-no-TZ**() devuelve 2014-01-15T14:00:00

current-date-no-TZ [altova:]

altova:current-date-no-TZ() como **xs:date** **XP3** **XQ3**
Esta función no toma ningún argumento. Quita la parte de zona horaria de la función `current-date()` (que es la fecha actual según el reloj del sistema) y devuelve un valor de tipo `xs:date`.

Ejemplos
Si la fecha actual es **2014-01-15+01:00**:
- **altova:current-date-no-TZ()** devuelve `2014-01-15`

current-time-no-TZ [altova:]

altova:current-time-no-TZ() as **xs:time** **XP3** **XQ3**
Esta función no toma ningún argumento. Quita la parte de zona horaria de `current-time()` (que es la hora actual según el reloj del sistema) y devuelve un valor de tipo `xs:time`.

Ejemplos
Si la hora actual es **14:00:00+01:00**:
- **altova:current-time-no-TZ()** devuelve `14:00:00`

[Subir]

Obtener el día de la semana de xs:dateTime o xs:date XP3 XQ3

Estas funciones obtienen el día de la semana (como entero) de `xs:dateTime` o `xs:date`. Los días de la semana se numeran del `1` al `7` (usando el formato EE UU, es decir `Domingo =1`). En el formato europeo la semana empieza el lunes (es decir, `Lunes=1`). Para establecer el formato EE UU (`Domingo=1`) use el entero `0` allí donde se acepte un entero para indicar el formato.

weekday-from-dateTime [altova:]

altova:weekday-from-dateTime(FechaHora *as xs:dateTime***)** como **xs:integer** **XP3** **XQ3**
Toma una fecha con hora como único argumento y devuelve el día de la semana de la fecha dada como número entero. Los días de la semana se numeran del 1 al 7 empezando por `Domingo=1`. Si necesita usar el formato europeo (donde `Lunes=1`), utilice la otra firma de esta función (*ver más abajo*).

Ejemplos
- **altova:weekday-from-dateTime**`(xs:dateTime("2014-02-03T09:00:00"))` devuelve `2`, lo cual significa "Lunes".

altova:weekday-from-dateTime(DateTime *as xs:dateTime*, **Formato** *as xs:integer***)** como **xs:integer** **XP3** **XQ3**
Toma una fecha con hora como primer argumento y devuelve el día de la semana de la fecha dada como número entero. Los días de la semana se numeran del 1 al 7 empezando por `Lunes=1`. Si el segundo argumento (**Formato**) es `0`, entonces los días de la semana se numeran del 1 al 7 empezando por `Domingo=1`. Si el segundo argumento es un entero

 © 2014 Altova GmbH

distinto de `0`, entonces `Lunes=1`. Si falta el segundo argumento, la función se lee como en la firma anterior (*ver más arriba*).

Ejemplos

- **altova:weekday-from-dateTime**(xs:dateTime("2014-02-03T09:00:00"), 1) devuelve 1, lo cual significa "Lunes".
- **altova:weekday-from-dateTime**(xs:dateTime("2014-02-03T09:00:00"), 4) devuelve 1, lo cual significa "Lunes".
- **altova:weekday-from-dateTime**(xs:dateTime("2014-02-03T09:00:00"), 0) devuelve 2, lo cual significa "Lunes".

weekday-from-date [altova:]

altova:weekday-from-date(Fecha *as xs:date*) como **xs:integer** **XP3** **XQ3**

Toma una fecha como único argumento y devuelve el día de la semana de la fecha dada como número entero. Los días de la semana se numeran del 1 al 7 empezando por `Domingo=1`. Si necesita usar el formato europeo (donde `Lunes=1`), utilice la otra firma de esta función (*ver más abajo*).

Ejemplos

- **altova:weekday-from-date**(xs:date("2014-02-03+01:00")) devuelve 2, lo cual significa "Lunes".

altova:weekday-from-date(Fecha *as xs:date*, **Formato** *as xs:integer*) como **xs:integer** **XP3** **XQ3**

Toma una fecha como primer argumento y devuelve el día de la semana de la fecha dada como número entero. Los días de la semana se numeran del 1 al 7 empezando por `Lunes=1`. Si el segundo argumento (**Formato**) es `0`, entonces los días de la semana se numeran del 1 al 7 empezando por `Domingo=1`. Si el segundo argumento es un entero distinto de `0`, entonces `Lunes=1`. Si falta el segundo argumento, la función se lee como en la firma anterior (*ver más arriba*).

Ejemplos

- **altova:weekday-from-date**(xs:date("2014-02-03"), 1) devuelve 1, lo cual significa "Lunes"
- **altova:weekday-from-date**(xs:date("2014-02-03"), 4) devuelve 1, lo cual significa "Lunes"
- **altova:weekday-from-date**(xs:date("2014-02-03"), 0) devuelve 2, lo cual significa "Lunes"

[Subir]

Generar tipos de datos de fecha, hora y duración a partir de sus componentes léxicos

XP3 **XQ3**

Estas funciones toman los componentes léxicos de los tipos de datos `xs:date`, `xs:time` y `xs:duration` como argumentos de entrada y los combinan para generar el tipo de datos correspondiente.

build-date [altova:]

altova:build-date(Año *as xs:integer*, **Mes** *as xs:integer*, **Fecha** *as xs:integer*)

como **xs:date** **XP3** **XQ3**
Los argumentos son el año, el mes y la fecha respectivamente. Se combinan para generar un valor de tipo `xs:date`. Los valores de los enteros deben estar en el intervalo de esa fecha en particular. Por ejemplo, el segundo argumento (para el mes) no puede ser mayor que `12`.

Ejemplos

- **altova:build-date**(2014, 2, 03) devuelve 2014-02-03

build-time [altova:]

altova:build-time(Horas *as xs:integer*, **Minutos** *as xs:integer*, **Segundos** *as xs:integer*) como **xs:time** **XP3** **XQ3**
El primer, segundo y tercer argumentos son la hora (`0` - `23`), los minutos (`0` - `59`) y los segundos (`0` - `59`) respectivamente. Se combinan para generar un valor de tipo `xs:time`. Los valores de los enteros deben estar dentro del intervalo correcto de esa parte de tiempo concreta. Por ejemplo, el segundo argumento (`Minutos`) no puede ser mayor que `59`. Para añadir la parte de uso horario al valor, use la firma que aparece más abajo.

Ejemplos

- **altova:build-time**(23, 4, 57) devuelve 23:04:57

altova:build-time(Horas *as xs:integer*, **Minutos** *as xs:integer*, **Segundos** *as xs:integer*, **UsoHorario** *as xs:string*) como **xs:time** **XP3** **XQ3**
The first, second, and third arguments are, respectively, the hour (`0` to `23`), minutes (`0` to `59`), and seconds (`0` to `59`) values. The fourth argument is a string that provides the timezone part of the value. The four arguments are combined to build a value of `xs:time` type. The values of the integers must be within the correct range of that particular time part. For example, the second (`Minutes`) argument should not be greater than `59`.

El primer, segundo y tercer argumentos son la hora (`0` - `23`), los minutos (`0` - `59`) y los segundos (`0` - `59`) respectivamente. El cuarto argumento es una cadena que aporta la parte de uso horario del valor. Los cuatro argumentos se combinan para generar un valor de tipo `xs:time`. Los valores de los enteros deben estar dentro del intervalo correcto de esa parte de tiempo concreta. Por ejemplo, el segundo argumento (`Minutos`) no puede ser mayor que `59`.

Ejemplos

- **altova:build-time**(23, 4, 57, '+1') devuelve 23:04:57+01:00

build-duration [altova:]

altova:build-duration(Años *as xs:integer*, **Meses** *as xs:integer*) como **xs:yearMonthDuration** **XP3** **XQ3**
Toma dos argumentos para generar un valor de tipo `xs:yearMonthDuration`. El primer argumento da la parte `Years` del valor de duración, mientras que el segundo da la parte `Months`. Si el segundo (`Months`) es mayor o igual que `12`, el entero se divide por 12. El cociente se añade al primer argumento para aportar la parte `Years` del valor de duración, mientras que el resto (de la división) da la parte `Months`. Para generar una duración de tipo `xs:dayTimeDuration`, consulte la firma siguiente.

Ejemplos

- **altova:build-duration**(2, 10) devuelve P2Y10M
- **altova:build-duration**(14, 27) devuelve P16Y3M
- **altova:build-duration**(2, 24) devuelve P4Y

 © 2014 Altova GmbH

altova:build-duration(Días *as xs:integer*, **Horas** *as xs:integer*, **Minutos** *as xs:integer*, **Segundos** *as xs:integer*) como **xs:dayTimeDuration** **XP3** **XQ3**

Toma cuatro argumentos y los combina para generar un valor de tipo `xs:dayTimeDuration`. El primer argumento da la parte `Days` del valor de duración, el segundo, el tercero y el cuarto dan las partes `Hours`, `Minutes` y `Seconds` respectivamente. Los tres argumentos de tiempo se convierten a un valor equivalente en cuanto a la unidad mayor siguiente y el resultado se utiliza para calcular el valor total de la duración. Por ejemplo, 72 segundos se convierte en `1M+12S` (1 minuto y 12 segundos) y este valor se usa para calcular el valor total de la duración. Para generar una duración de tipo `xs:yearMonthDuration`, consulte la firma anterior.

Ejemplos

- **altova:build-duration**(2, 10, 3, 56) devuelve P2DT10H3M56S
- **altova:build-duration**(1, 0, 100, 0) devuelve P1DT1H40M
- **altova:build-duration**(1, 0, 0, 3600) devuelve P1DT1H

[Subir]

Funciones para calcular la edad **XP3** **XQ3**

Estas funciones devuelven la edad que se calcula obteniendo la diferencia (i) entre la fecha del argumento de entrada y la fecha actual o (ii) entre las fechas de los dos argumentos de entrada. La función **altova:age** devuelve la edad en años, mientras que la función **altova:age-details** devuelve la edad en forma de una secuencia de tres enteros (años, meses y días).

age [altova:]

altova:age(FechaInicio *as xs:date*) como **xs:integer** **XP3** **XQ3**

Devuelve un entero que es la edad *en años* de algún objeto, contando a partir de la fecha de inicio dada como argumento y hasta la fecha actual (tomada del reloj del sistema). Si el argumento de entrada es un año o más después que la fecha actual, el valor devuelto será negativo.

Ejemplos

Si la fecha actual es **2014-01-15**:

- **altova:age**(xs:date("2013-01-15")) devuelve 1
- **altova:age**(xs:date("2013-01-16")) devuelve 0
- **altova:age**(xs:date("2015-01-15")) devuelve -1
- **altova:age**(xs:date("2015-01-14")) devuelve 0

altova:age(FechaInicio *as xs:date*, **FechaFinal** *as xs:date*) como **xs:integer** **XP3** **XQ3**

Devuelve un entero que es la edad *en años* de algún objeto, contando a partir de la fecha de inicio dada como primer argumento y hasta la fecha dada como segundo argumento. El valor devuelto será negativo si el primer argumento es un año o más después que el segundo argumento.

Ejemplos

- **altova:age**(xs:date("2000-01-15"), xs:date("2010-01-15")) devuelve 10
- **altova:age**(xs:date("2000-01-15"), current-date()) devuelve 14 si la fecha actual es 2014-01-15

- `altova:age(xs:date("2014-01-15"), xs:date("2010-01-15"))` devuelve `-4`

age-details [altova:]

`altova:age-details(FechaEntrada as xs:date)` como `(xs:integer)*` **XP3 XQ3**

Devuelve tres enteros que son los años, meses y días respectivamente que hay entre la fecha dada como argumento y la fecha actual (tomada del reloj del sistema). La suma del valor devuelto nos da el tiempo total transcurrido entre ambas fechas (entre la fecha dada y la fecha actual). La fecha de entrada puede tener un valor anterior o posterior a la fecha actual, pero esto no se indica en el valor devuelto por medio de un signo negativo o positivo. El valor devuelto siempre es positivo.

Ejemplos

Si la fecha actual es **2014-01-15**:

- `altova:age-details(xs:date("2014-01-16"))` devuelve `(0 0 1)`
- `altova:age-details(xs:date("2014-01-14"))` devuelve `(0 0 1)`
- `altova:age-details(xs:date("2013-01-16"))` devuelve `(1 0 1)`
- `altova:age-details(current-date())` devuelve `(0 0 0`

`altova:age-details(Fecha1 as xs:date, Fecha2 as xs:date)` como `(xs:integer)*` **XP3 XQ3**

Devuelve tres enteros que son los años, meses y días que hay entre las dos fechas dadas por los argumentos. La suma del valor devuelto nos da el tiempo total transcurrido entre las dos fechas de entrada. Da igual cuál de las dos fechas se da como primer argumento, la más antigua o la más reciente. El valor devuelto no indica si la fecha de entrada es anterior o posterior a la fecha actual. Es decir, el valor devuelto siempre es positivo.

Ejemplos

- `altova:age-details(xs:date("2014-01-16"), xs:date("2014-01-15"))` devuelve `(0 0 1)`
- `altova:age-details(xs:date("2014-01-15"), xs:date("2014-01-16"))` devuelve `(0 0 1)`

[Subir]

10.1.3 Funciones XPath/XQuery: funciones de cadena

Estas funciones de extensión XPath/XQuery generales son compatibles con la versión actual de su producto de Altova y se pueden usar en (i) expresiones XPath en contextos XSLT o en (ii) expresiones XQuery en documentos XQuery.

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. Tenga en cuenta que en futuras versiones del producto algunas funciones pueden dejar de ser compatibles o su comportamiento puede cambiar. Por tanto, consulte siempre la documentación del producto para conocer el funcionamiento de estas funciones en cada versión del producto.

Funciones XPath (en expresiones XPath en XSLT):	XP1 XP2 XP3
Funciones XSLT (en expresiones XPath en XSLT):	XSLT1 XSLT2 XSLT3
Funciones XQuery (en expresiones XQuery en XQuery):	XQ1 XQ3

---

char [altova:]

altova:char(Posición *as xs:integer*) como **xs:string** XP3 XQ3

Devuelve una cadena que contiene el carácter que está en la posición indicada por el argumento **Posición** en la cadena que se obtiene al convertir el valor del elemento de contexto en `xs:string`. La cadena resultante estará vacía si en la posición indicada no existe ningún carácter.

Ejemplos

Si el elemento de contexto es **1234ABCD**:

- `altova:char(2)` devuelve `2`
- `altova:char(5)` devuelve `A`
- `altova:char(9)` devuelve la cadena vacía
- `altova:char(-2)` devuelve la cadena vacía

altova:char(CadenaEntrada *as xs:string*, **Posición** *as xs:integer*) como **xs:string** XP3 XQ3

Devuelve una cadena que contiene el carácter que está en la posición indicada por el argumento **Posición** en la cadena dada por el argumento **CadenaEntrada**. La cadena resultante estará vacía si en la posición indicada no existe ningún carácter.

Ejemplos

- `altova:char("2014-01-15", 5)` devuelve `-`
- `altova:char("USA", 1)` devuelve `U`
- `altova:char("USA", 1)` devuelve la cadena vacía
- `altova:char("USA", -2)` devuelve la cadena vacía

first-chars [altova:]

altova:first-chars(X *as xs:integer*) como **xs:string** **XP3** **XQ3**
Devuelve una cadena que contiene los **x** primeros caracteres de la cadena que se obtiene al convertir el valor del elemento de contexto en `xs:string`.

Ejemplos
Si el elemento de contexto es **1234ABCD**:
- **altova:first-chars**(2) devuelve 12
- **altova:first-chars**(5) devuelve 1234A
- **altova:first-chars**(9) devuelve 1234ABCD

altova:first-chars(CadenaEntrada *as xs:string*, **X** *as xs:integer*) como **xs:string** **XP3** **XQ3**
Devuelve una cadena que contiene los **x** primeros caracteres de la cadena dada como argumento **CadenaEntrada**.

Ejemplos
- **altova:first-chars**("2014-01-15", 5) devuelve 2014-
- **altova:first-chars**("USA", 1) devuelve U

last-chars [altova:]

altova:last-chars(X *as xs:integer*) como **xs:string** **XP3** **XQ3**
Devuelve una cadena que contiene los X últimos caracteres de la cadena que se obtiene al convertir el valor del elemento de contexto en `xs:string`.

Ejemplos
Si el elemento de contexto es **1234ABCD**:
- **altova:last-chars**(2) devuelve CD
- **altova:last-chars**(5) devuelve 4ABCD
- **altova:last-chars**(9) devuelve 1234ABCD

altova:last-chars(CadenaEntrada *as xs:string*, **X** *as xs:integer*) como **xs:string** **XP3** **XQ3**
Devuelve una cadena que contiene los **x** últimos caracteres de la cadena dada como argumento **CadenaEntrada**.

Ejemplos
- **altova:last-chars**("2014-01-15", 5) devuelve 01-15-
- **altova:last-chars**("USA", 10) devuelve USA

pad-string-left [altova:]

altova:pad-string-left(Cadena *as xs:string*, **Repeticiones** *as xs:integer*, **CarácterRelleno** *as xs:string*) como **xs:string** **XP3** **XQ3**
El argumento **CarácterRelleno** es un solo carácter que se incluye a la izquierda de la cadena dada como argumento **Cadena**. El argumento **Repeticiones** da el número de veces que se debe repetir el carácter de relleno a la izquierda de **Cadena**.

Ejemplos
- **altova:pad-string-left**('AP', 1, 'Z') devuelve 'ZAP'
- **altova:pad-string-left**('AP', 3, 'Z') devuelve 'ZZZAP'
- **altova:pad-string-left**('AP', 0, 'Z') devuelve 'AP'
- **altova:pad-string-left**('AP', 3, 'YZ') devuelve el error "Carácter de relleno

demasiado largo".

pad-string-right [altova:]

altova:pad-string-right(Cadena *as xs:string*, **Repeticiones** *as xs:integer*, **CarácterRelleno** *as xs:string*) cpmo **xs:string** **XP3** **XQ3**

El argumento **CarácterRelleno** es un solo carácter que se incluye a la derecha de la cadena dada como argumento **Cadena**. El argumento **Repeticiones** da el número de veces que se debe repetir el carácter de relleno a la derecha de **Cadena**.

Ejemplos

- **altova:pad-string-right**('AP', 1, 'Z') devuelve 'APZ'
- **altova:pad-string-right**('AP', 3, 'Z') devuelve 'APZZZ'
- **altova:pad-string-right**('AP', 0, 'Z') devuelve 'AP'
- **altova:pad-string-right**('AP', 3, 'YZ') devuelve el error "Carácter de relleno demasiado largo".

repeat-string [altova:]

altova:repeat-string(CadenaEntrada *as xs:string*, **Repeticiones** *as xs:integer*) como **xs:string** **XP2** **XQ1** **XP3** **XQ3**

Genera una cadena que está compuesta por el primer argumento `CadenaEntrada` repetida tantas veces como indique el argumento `Repeticiones`.

Ejemplo

- **altova:repeat-string**("Altova #", 3)

  devuelve `Altova #Altova #Altova #`"

substring-after-last [altova:]

altova:substring-after-last(CadenaPrincipal *as xs:string*, **CadenaPrueba** *as xs:string*) como **xs:string** **XP3** **XQ3**

Si **CadenaPrueba** se encuentra en **CadenaPrincipal**, la función devuelve la subcadena que aparece después de **CadenaPrueba** en **CadenaPrincipal**. Si **CadenaPrueba** no está en **CadenaPrincipal**, entonces devuelve la cadena vacía. Si **CadenaPrueba** es una cadena vacía, entonces devuelve la **CadenaPrincipal** entera. Si **CadenaPrueba** aparece varias veces en **CadenaPrincipal**, la función devuelve la subcadena que aparece después de la última **CadenaPrueba**.

Ejemplos

- **altova:substring-after-last**('ABCDEFGH', 'B') devuelve 'CDEFGH'
- **altova:substring-after-last**('ABCDEFGH', 'BC') devuelve 'DEFGH'
- **altova:substring-after-last**('ABCDEFGH', 'BD') devuelve ''
- **altova:substring-after-last**('ABCDEFGH', 'Z') devuelve ''
- **altova:substring-after-last**('ABCDEFGH', '') devuelve 'ABCDEFGH'
- **altova:substring-after-last**('ABCD-ABCD', 'B') devuelve 'CD'
- **altova:substring-after-last**('ABCD-ABCD-ABCD', 'BCD') devuelve ''

substring-before-last [altova:]

altova:substring-before-last(CadenaPrincipal *as xs:string*, **CadenaPrueba** *as*

xs:string) como **xs:string** XP3 XQ3

Si **CadenaPrueba** se encuentra en **CadenaPrincipal**, la función devuelve la subcadena que aparece después de **CadenaPrueba** en **CadenaPrincipal**. Si **CadenaPrueba** no está en **CadenaPrincipal**, entonces devuelve la cadena vacía. Si **CadenaPrueba** es una cadena vacía, entonces devuelve la **CadenaPrincipal** entera. Si **CadenaPrueba** aparece varias veces en **CadenaPrincipal**, la función devuelve la subcadena que aparece antes de la última **CadenaPrueba**.

Ejemplos

- **altova:substring-before-last**('ABCDEFGH', 'B') devuelve 'A'
- **altova:substring-before-last**('ABCDEFGH', 'BC') devuelve 'A'
- **altova:substring-before-last**('ABCDEFGH', 'BD') devuelve ''
- **altova:substring-before-last**('ABCDEFGH', 'Z') devuelve ''
- **altova:substring-before-last**('ABCDEFGH', '') devuelve ''
- **altova:substring-before-last**('ABCD-ABCD', 'B') devuelve 'ABCD-A'
- **altova:substring-before-last**('ABCD-ABCD-ABCD', 'ABCD') devuelve 'ABCD-ABCD-'

substring-pos [altova:]

altova:substring-pos(**Cadena** *as xs:string*, **CadenaBúsqueda** *as xs:string*) como **xs:integer** XP3 XQ3

Devuelve la posición de carácter de la primera instancia de **CadenaBúsqueda** en **Cadena**. La posición de carácter se devuelve como número entero. El primer carácter de **CadenaBúsqueda** tiene la posición 1. Si **CadenaBúsqueda** no aparece dentro de **Cadena**, la función devuelve el entero 0. Para buscar la segunda instancia de **CadenaBúsqueda**, etc. use la otra firma de esta función.

Ejemplos

- **altova:substring-pos**('Altova', 'to') devuelve 3
- **altova:substring-pos**('Altova', 'tov') devuelve 3
- **altova:substring-pos**('Altova', 'tv') devuelve 0
- **altova:substring-pos**('AltovaAltova', 'to') devuelve 3

altova:substring-pos(**Cadena** *as xs:string*, **CadenaBúsqueda** *as xs:string*, **Entero** *as xs:integer*) como **xs:integer** XP3 XQ3

Devuelve la posición de carácter de **CadenaBúsqueda** en **Cadena**. La búsqueda de **CadenaBúsqueda** empieza en la posición de carácter dada por el argumento **Entero** (es decir, no se busca en la subcadena anterior a esta posición). El entero devuelto, sin embargo, es la posición que la cadena encontrada tiene en **Cadena**. Esta firma es muy práctica si quiere buscar la segunda posición, etc. de una cadena que aparece varias veces dentro de **Cadena**. Si **CadenaBúsqueda** no aparece en **Cadena**, la función devuelve el entero 0.

Ejemplos

- **altova:substring-pos**('Altova', 'to', 1) devuelve 3
- **altova:substring-pos**('Altova', 'to', 3) devuelve 3
- **altova:substring-pos**('Altova', 'to', 4) devuelve 0
- **altova:substring-pos**('Altova-Altova', 'to', 0) devuelve 3
- **altova:substring-pos**('Altova-Altova', 'to', 4) devuelve 10

10.1.4 Funciones XPath/XQuery: funciones varias

Estas funciones de extensión XPath/XQuery generales son compatibles con la versión actual de su producto de Altova y se pueden usar en (i) expresiones XPath en contextos XSLT o (ii) en expresiones XQuery en documentos XQuery.

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. Tenga en cuenta que en futuras versiones del producto algunas funciones pueden dejar de ser compatibles o su comportamiento puede cambiar. Por tanto, consulte siempre la documentación del producto para conocer el funcionamiento de estas funciones en cada versión del producto.

Funciones XPath (en expresiones XPath en XSLT):	XP1 XP2 XP3
Funciones XSLT (en expresiones XPath en XSLT):	XSLT1 XSLT2 XSLT3
Funciones XQuery (en expresiones XQuery en XQuery):	XQ1 XQ3

Funciones de numeración automática

generate-auto-number [altova:]

altova:generate-auto-number(ID *as xs:string*, **StartsWith** *as xs:double*, **Increment** *as xs:double*, **ResetOnChange** *as xs:string*) como `xs:integer` XP1 XP2 XQ1 XP3 XQ3

Genera un número cada vez que se llama a la función. El primer número, que se genera cuando se llama a la función por primera vez, viene dado por el argumento `StartsWith`. Cada llamada posterior genera un número nuevo, que se incrementa en función del valor especificado en el argumento `Increment`. De hecho, la función `altova:generate-auto-number` crea un contador llamado como indique el argumento `ID` y este contador se incrementa cada vez que se llama a la función. Si el valor del argumento `ResetOnChange` cambia con respecto al valor que tenía en la llamada anterior, entonces el valor del número que se debe generar se restablece con el valor de `StartsWith`. También puede restablecer la numeración automática con la función `altova:reset-auto-number`.

Ejemplo

- **altova:generate-auto-number**("ChapterNumber", 1, 1, "SomeString")

Devuelve un número cada vez que se llama a la función, empezando por `1` y con un incremento de `1` con cada llamada a función. Si el cuarto argumento continúa siendo `"SomeString"` en las llamadas posteriores, el incremento continuará. Cuando cambie el valor del cuarto argumento, se restaura el valor 1 del contador (llamado `ChapterNumber`). El valor de `ChapterNumber` también se puede restaurar llamando a la función `altova:reset-auto-number`: `altova:reset-auto-number("ChapterNumber")`.

reset-auto-number [altova:]

altova:reset-auto-number(ID *as xs:string*) XP1 XP2 XQ1 XP3 XQ3

Esta función restaura el número del contador de numeración automática especificado en el argumento `ID`. El número se reemplaza con el número indicado en el argumento `StartsWith` de la función `altova:generate-auto-number` que creó el contador especificado en el argumento `ID`.

Ejemplos

- **`altova:reset-auto-number`**`("ChapterNumber")` restablece el número del contador de numeración automática llamado `ChapterNumber` que se creó con la función `altova:generate-auto-number`. El número se reemplaza con el valor del argumento `StartsWith` de la función `altova:generate-auto-number` que creó `ChapterNumber`.

[Subir]

Funciones numéricas

hex-string-to-integer [altova:]

`altova:hex-string-to-integer(CadenaHex` *`as xs:string`*`)` como **`xs:integer`** **XP3** **XQ3**

Toma un argumento de cadena que es el equivalente Base-16 de un entero del sistema decimal (Base-10) y devuelve un entero decimal.

Ejemplos

- **`altova:hex-string-to-integer`**`('1')` devuelve `1`
- **`altova:hex-string-to-integer`**`('9')` devuelve `9`
- **`altova:hex-string-to-integer`**`('A')` devuelve `10`
- **`altova:hex-string-to-integer`**`('B')` devuelve `11`
- **`altova:hex-string-to-integer`**`('F')` devuelve `15`
- **`altova:hex-string-to-integer`**`('G')` devuelve un error
- **`altova:hex-string-to-integer`**`('10')` devuelve `16`
- **`altova:hex-string-to-integer`**`('01')` devuelve `1`
- **`altova:hex-string-to-integer`**`('20')` devuelve `32`
- **`altova:hex-string-to-integer`**`('21')` devuelve `33`
- **`altova:hex-string-to-integer`**`('5A')` devuelve `90`
- **`altova:hex-string-to-integer`**`('USA')` devuelve un error

integer-to-hex-string [altova:]

`altova:integer-to-hex-string(Entero` *`as xs:integer`*`)` como **`xs:string`** **XP3** **XQ3**

Toma el argumento **`Entero`** y devuelve su equivalente Base-16 en forma de cadena.

Ejemplos

- **`altova:integer-to-hex-string`**`(1)` devuelve `'1'`
- **`altova:integer-to-hex-string`**`(9)` devuelve `'9'`
- **`altova:integer-to-hex-string`**`(10)` devuelve `'A'`
- **`altova:integer-to-hex-string`**`(11)` devuelve `'B'`
- **`altova:integer-to-hex-string`**`(15)` devuelve `'F'`
- **`altova:integer-to-hex-string`**`(16)` devuelve `'10'`
- **`altova:integer-to-hex-string`**`(32)` devuelve `'20'`
- **`altova:integer-to-hex-string`**`(33)` devuelve `'21'`
- **`altova:integer-to-hex-string`**`(90)` devuelve `'5A'`

 © 2014 Altova GmbH

[Subir]

Funciones de secuencia

attributes [altova:]

altova:attributes(NombreAtributo *as xs:string*) como **attribute()*** **XP3** **XQ3**
Devuelve todos los atributos cuyo nombre local coincida con el nombre dado como argumento de entrada (`NombreAtributo`). La búsqueda tiene en cuenta el uso de mayúsculas y minúsculas y se lleva a cabo en el eje `attribute::`.

Ejemplos

- **altova:attributes**("MiAtributo") devuelve `MiAtributo()*`

altova:attributes(NombreAtributo *as xs:string*, **OpcionesBúsqueda** *as xs:string*) como **attribute()*** **XP3** **XQ3**
Devuelve todos los atributos cuyo nombre local coincida con el nombre dado como argumento de entrada (`NombreAtributo`). La búsqueda tiene en cuenta el uso de mayúsculas y minúsculas y se lleva a cabo en el eje `attribute::`. El segundo argumento es una cadena con marcas de búsqueda. Estas son las marcas disponibles:

r = habilita la búsqueda de expresiones regulares. En este caso, `NombreAtributo` debe ser una cadena de búsqueda de expresión regular;
i = la búsqueda no tiene en cuenta el uso de mayúsculas y minúsculas;
p = incluye el prefijo de espacio de nombres en la búsqueda. En este caso, `NombreAtributo` debe contener el prefijo de espacio de nombres (p. ej.: `altova:MiAtributo`).

Las marcas pueden escribirse en cualquier orden y no hace falta utilizar todas. Si usa marcas no válidas, se genera un error. También puede usar una cadena vacía para el segundo argumento. Esto tiene el mismo efecto que usar solo el primer argumento. Sin embargo, no está permitido usar una secuencia vacía.

Ejemplos

- **altova:attributes**("MiAtributo", "rip") devuelve `MiAtributo()*`
- **altova:attributes**("MiAtributo", "pri") devuelve `MiAtributo()*`
- **altova:attributes**("MiAtributo", "") devuelve `MiAtributo()*`
- **altova:attributes**("MiAtributo", "Rip") devuelve un error de marca desconocida.
- **altova:attributes**("MiAtributo",) devuelve un error diciendo que falta el segundo argumento.

elements [altova:]

altova:elements(NombreElemento *as xs:string*) como **elemento()*** **XP3** **XQ3**
Devuelve todos los elementos cuyo nombre local coincida con el nombre dado como argumento de entrada (`NombreElemento`). La búsqueda tiene en cuenta el uso de mayúsculas y minúsculas y se lleva a cabo en el eje `child::`.

Ejemplos

- **altova:elements**("MiElemento") devuelve `MiElemento()*`

altova:elements(NombreElemento *as xs:string*, **OpcionesBúsqueda** *as xs:string*) como **elemento()*** **XP3** **XQ3**
Devuelve todos los elementos cuyo nombre local coincida con el nombre dado como

argumento de entrada (`NombreElemento`). La búsqueda tiene en cuenta el uso de mayúsculas y minúsculas y se lleva a cabo en el eje `child::`. El segundo argumento es una cadena con marcas de búsqueda. Estas son las marcas disponibles:

r = habilita la búsqueda de expresiones regulares. En este caso, `NombreElemento` debe ser una cadena de búsqueda de expresión regular;
i = la búsqueda no tiene en cuenta el uso de mayúsculas y minúsculas;
p = incluye el prefijo de espacio de nombres en la búsqueda. En este caso, `NombreElemento` debe contener el prefijo de espacio de nombres (p. ej.: `altova:MiElemento`).

Las marcas pueden escribirse en cualquier orden y no hace falta utilizar todas. Si usa marcas no válidas, se genera un error. También puede usar una cadena vacía para el segundo argumento. Esto tiene el mismo efecto que usar solo el primer argumento. Sin embargo, no está permitido usar una secuencia vacía.

Ejemplos

- **altova:elements**("MiElemento", "rip") devuelve `MiElemento()*`
- **altova:elements**("MiElemento", "pri") devuelve `MiElemento()*`
- **altova:elements**("MiElemento", "") devuelve `MiElemento()*`
- **altova:elements**("MiElemento", "Rip") devuelve un error de marca desconocida.
- **altova:elements**("MiElemento",) devuelve un error diciendo que falta el segundo argumento.

substitute-empty [altova:]

altova:substitute-empty(PrimeraSecuencia *as item()**, **SegundaSecuencia** *as item()***)** como **item()*** **XP3** **XQ3**

Si **PrimeraSecuencia** está vacío, la función devuelve **SegundaSecuencia**. Si **PrimeraSecuencia** no está vacío, la función devuelve **PrimeraSecuencia**.

Ejemplos

- **altova:substitute-empty**((1,2,3), (4,5,6)) devuelve (1,2,3)
- **altova:substitute-empty**((), (4,5,6)) devuelve (4,5,6)

[Subir]

Funciones para trabajar con identificadores URI

get-temp-folder [altova:]

altova:get-temp-folder() como `xs:string` **XP2** **XQ1** **XP3** **XQ3**

Esta función no toma ningún argumento. Devuelve la ruta de acceso de la carpeta temporal del usuario actual.

Ejemplo

- **altova:get-temp-folder**()
  en un equipo Windows devuelve (más o menos) `C:\Usuarios\<Usuario>\AppData\Local\Temp\` como valor de tipo `xs:string`.

 © 2014 Altova GmbH

[Subir]

10.1.5 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. No obstante, tenga en cuenta que en futuras versiones del producto algunas funciones pueden dejar de ser compatibles o su comportamiento puede cambiar. Por tanto, consulte siempre la documentación del producto para conocer el funcionamiento de estas funciones en cada versión del producto.

Las funciones para gráficos son **funciones XPath** (no XSLT) y las hay de dos tipos:

- 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 gráficos son `jpg`, `png` y `bmp`. La opción más recomendable es `png` porque no pierde información y es un formato comprimido. En las ediciones **Enterprise Edition** de las herramientas de escritorio de Altova, los formatos compatibles son `jpg, png, bmp` y `gif`.

Funciones para generar y guardar gráficos
Estas funciones toman el objeto del gráfico (obtenido con las funciones de creación de gráficos) y generan una imagen o guardan una imagen en un archivo.

`altova:generate-chart-image` `($chart, $width, $height, $encoding) as atomic`

donde

- `$chart` es el componente de extensión de gráficos obtenido con la función `altova:create-chart`
- `$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 `base64Binary` o `hexBinary`
- `$imagetype` puede ser uno de estos formatos de imagen: `png`, `gif`, `bmp`, `jpg`, `jpeg`.

 © 2014 Altova GmbH

Recuerde que el formato `gif` no es compatible con los productos servidor de Altova (*ver nota al principio de este apartado*)

La función devuelve la imagen del gráfico en la codificación y formato de imagen elegidos.

`altova:save-chart-image` `($chart, $filename, $width, $height) as empty()` ***(solo en Windows)***

donde

- `$chart` es el componente de extensión de gráficos obtenido con la función `altova:create-chart`
- `$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 de gráficos 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 de gráficos chart-config

donde

- $xml-struct es la estructura XML que contiene los datos de configuración del gráfico

La función devuelve un componente de extensión chart-config que contiene los datos de configuración del gráfico. Estos datos se suministran en un fragmento de código XML.

---

altova:create-chart-data-series($series-name?, $x-values*, $y-values*) como componente de extensión de gráficos chart-data-series

donde

- $series-name especifica el nombre de la serie
- $x-values presenta la lista de valores del eje X
- $y-values presenta la lista de valores del eje Y

La función devuelve un componente de extensión chart-data-series que contiene los datos necesarios para generar el gráfico: es decir, el nombre de las series y los datos de los ejes.

 © 2014 Altova GmbH

---

altova:create-chart-data-row(x, y1, y2, y3, ...) como componente de extensión de gráficos chart-data-x-*N*y-row

donde

- x es el valor de la columna del eje X de la fila de datos del gráfico
- yN son los valores de las columnas del eje Y

La función devuelve un componente de extensión chart-data-x-Ny-row, que contiene los datos para la columna del eje X y las columnas del eje Y de una sola serie.

---

altova:create-chart-data-series-from-rows($series-names as xs:string*, $row*) como componente de extensión de gráficos chart-data-series

donde

- $series-name es el nombre de la series que se debe crear
- $row es el componente de extensión chart-data-x-Ny-row que se debe crear como serie

La función devuelve un componente de extensión chart-data-series, que contiene los datos para el eje X y el eje Y de la serie.

---

altova:create-chart-layer($chart-config, $chart-data-series*) como componente de extensión de gráficos chart-layer

donde

- $chart-config es el componente de extensión chart-config obtenido con la función 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 de gráficos multicapas obtenido con la función `altova:create-chart-layer`

La función devuelve un componente de extensión de gráficos multicapas.

---

altova:create-multi-layer-chart`($chart-config, $chart-data-series*, $chart-layer*, xs:boolean $mergecategoryvalues)`

donde

- `$chart-config` es el componente de extensión chart-config obtenido con la función `altova:create-chart-config` o con la función `altova:create-chart-config-from-xml`
- `$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 de gráficos multicapa obtenido con la función `altova:create-chart-layer`

La función devuelve un componente de extensión de gráficos multicapa.

Estructura XML de los datos de gráficos

A continuación puede ver un fragmento de código XML con datos de gráfico, tal y como aparecería para las funciones de extensión para gráficos de Altova. 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>

    <Line
        ConnectionShapeSize="1.%"   PorcentajeOPixel
        DrawFilledConnectionShapes="1"   Bool
        DrawOutlineConnectionShapes="0"   Bool
        DrawSlashConnectionShapes="0"   Bool
        DrawBackslashConnectionShapes="0"   Bool
    />

    <Bar
        ShowShadow="1"   Bool
        ShadowColor="#a0a0a0"   Color
        OutlineColor="#000000"   Color
        ShowOutline="1"   Bool
    />
```

<Area
`Transparency="0"` *UINT (0-255) 255 es totalmente transparente y 0 es opaco*
`OutlineColor="#000000"` *Color*
`ShowOutline="1"` *Bool*
/>

<CandleStick
`FillHighClose="0"` *Bool. Si es 0, el cuerpo está vacío. Si es 1, FillColorHighClose se usa para el cuerpo de la vela*
`FillColorHighClose="#ffffff"` *Color. Para el cuerpo de la vela cuando el valor de close > que el valor de open*
`FillHighOpenWithSeriesColor="1"` *Bool. Si es true, el color de la serie se usa para rellenar el cuerpo de la vela cuando el valor de open > que el valor de close*
`FillColorHighOpen="#000000"` *Color. Para el cuerpo de la vela cuando el valor de open > que el valor de close y* `FillHighOpenWithSeriesColor` *es false*
/>

<Colors *Combinación de colores definida por el usuario. Este elemento está vacío por defecto, excepto el estilo, y no tiene atributos Color*
`UseSubsequentColors ="1"` *Booleano. Si es* `0`*, entonces se superpone el color. Si es* `1`*, se usan los colores siguientes de la capa de gráfico anterior*
`Style="User"` *Valores posibles: "Default", "Grayscale", "Colorful", "Pastel", "User"*
`Colors="#52aca0"` *Color: solamente se añade para el conjunto de colores definido por el usuario*
`Colors1="#d3c15d"` *Color: solamente se añade para el conjunto de colores definido por el usuario*
`Colors2="#8971d8"` *Color: solamente se añade para el conjunto de colores definido por el usuario*
`...`
`ColorsN=""` *Cada conjunto de colores puede tener un máximo de diez colores, de* `Colors` *a* `Colors9`
</Colors>

<Pie
`ShowLabels="1"` *Bool*
`OutlineColor="#404040"` *Color*
`ShowOutline="1"` *Bool*
`StartAngle="0."` *Double*
`Clockwise="1"` *Bool*
`Draw2dHighlights="1"` *Bool*
`Transparency="0"` *Int (De 0 a 255: 0 es opaco, 255 es totalmente transparente)*
`DropShadowColor="#c0c0c0"` *Color*
`DropShadowSize="5.%"` *PorcentajeOPíxel*
`PieHeight="10.%"` *PorcentajeOPíxel. Los valores de píxel pueden ser diferente en el resultado debido a la inclinación 3D*
`Tilt="40.0"` *Double (De 10 a 90: la inclinación 3D en grados de un gráfico circular 3D)*
`ShowDropShadow="1"` *Bool*
`ChartToLabelMargin="10.%"` *PorcentajeOPíxel*
`AddValueToLabel="0"` *Bool*
`AddPercentToLabel="0"` *Bool*
`AddPercentToLabels_DecimalDigits="0"` *UINT (0 – 2)*
>
`<LabelFont`

```
        Color="#000000"
        Name="Arial"
        Bold="0"
        Italic="0"
        Underline="0"
        MinFontHeight="10.pt"
        Size="4.%"/>
</Pie>

<XY>
    <XAxis   Axis
        AutoRange="1"   Bool
        AutoRangeIncludesZero="1"   Bool
        RangeFrom="0."   Double: intervalo manual
        RangeTill="1."   Double : intervalo manual
        LabelToAxisMargin="3.%"   PorcentajeOPíxel
        AxisLabel=""   Cadena de texto
        AxisColor="#000000"   Color
        AxisGridColor="#e6e6e6"   Color
        ShowGrid="1"   Bool
        UseAutoTick="1"   Bool
        ManualTickInterval="1."   Double
        AxisToChartMargin="0.px"   PorcentajeOPíxel
        TickSize="3.px"   PorcentajeOPíxel
        ShowTicks="1"   Bool
        ShowValues="1"   Bool
        AxisPosition="LeftOrBottom"   Enums: "LeftOrBottom", "RightOrTop", "AtValue"
        AxisPositionAtValue = "0"   Double
        >
        <ValueFont
            Color="#000000"
            Name="Tahoma"
            Bold="0"
            Italic="0"
            Underline="0"
            MinFontHeight="10.pt"
            Size="3.%"/>
    </XAxis>
    <YAxis   Eje (igual que XAxis)
        AutoRange="1"
        AutoRangeIncludesZero="1"
        RangeFrom="0."
        RangeTill="1."
        LabelToAxisMargin="3.%"
        AxisLabel=""
        AxisColor="#000000"
        AxisGridColor="#e6e6e6"
        ShowGrid="1"
        UseAutoTick="1"
        ManualTickInterval="1."
        AxisToChartMargin="0.px"
        TickSize="3.px"
        ShowTicks="1"   Bool
        ShowValues="1"   Bool
        AxisPosition="LeftOrBottom"   Enums: "LeftOrBottom", "RightOrTop", "AtValue"
```

```
            AxisPositionAtValue = "0"   Double
            >
            <ValueFont
                Color="#000000"
                Name="Tahoma"
                Bold="0"
                Italic="0"
                Underline="0"
                MinFontHeight="10.pt"
                Size="3.%"/>
        </YAxis>
    </XY>

    <XY3d
        AxisAutoSize="1"   Bool: Si es false, XSize y YSize definen la relación de aspecto de
los ejes x e y. Si es true, la relación de aspecto es igual a la ventana del gráfico
        XSize="100.%"   PorcentajeOPíxel. Los valores en píxel pueden ser diferentes en el
resultado debido a la inclinación 3D y a la opción de ajustar al tamaño
        YSize="100.%"   PorcentajeOPíxel. Los valores en píxel pueden ser diferentes en el
resultado debido a la inclinación 3D y a la opción de ajustar al tamaño
        SeriesMargin="30.%"   PorcentajeOPíxel. Los valores en píxel pueden ser diferentes en
el resultado debido a la inclinación 3D y a la opción de ajustar al tamaño
        Tilt="20."   Double. De -90 a +90 grados
        Rot="20."   Double. De -359 a +359 grados
        FoV="50.">   Double. Campo de visión: de 1 a 120 grados
        >
        <ZAxis
            AutoRange="1"
            AutoRangeIncludesZero="1"
            RangeFrom="0."
            RangeTill="1."
            LabelToAxisMargin="3.%"
            AxisLabel=""
            AxisColor="#000000"
            AxisGridColor="#e6e6e6"
            ShowGrid="1"
            UseAutoTick="1"
            ManualTickInterval="1."
            AxisToChartMargin="0.px"
            TickSize="3.px" >
            <ValueFont
                Color="#000000"
                Name="Tahoma"
                Bold="0"
                Italic="0"
                Underline="0"
                MinFontHeight="10.pt"
                Size="3.%"/>
        </ZAxis>
    </XY3d>

    <Gauge
        MinVal="0."   Double
        MaxVal="100."   Double
        MinAngle="225"   UINT: -359-359
```

```
        SweepAngle="270"   UINT: 1-359
        BorderToTick="1.%"   PorcentajeOPíxel
        MajorTickWidth="3.px"   PorcentajeOPíxel
        MajorTickLength="4.%"   PorcentajeOPíxel
        MinorTickWidth="1.px"   PorcentajeOPíxel
        MinorTickLength="3.%"   PorcentajeOPíxel
        BorderColor="#a0a0a0"   Color
        FillColor="#303535"   Color
        MajorTickColor="#a0c0b0"   Color
        MinorTickColor="#a0c0b0"   Color
        BorderWidth="2.%"   PorcentajeOPíxel
        NeedleBaseWidth="1.5%"   PorcentajeOPíxel
        NeedleBaseRadius="5.%"   PorcentajeOPíxel
        NeedleColor="#f00000"   Color
        NeedleBaseColor="#141414"   Color
        TickToTickValueMargin="5.%"   PorcentajeOPíxel
        MajorTickStep="10."   Double
        MinorTickStep="5."   Double
        RoundGaugeBorderToColorRange="0.%"   PorcentajeOPíxel
        RoundGaugeColorRangeWidth ="6.%"   PorcentajeOPíxel
        BarGaugeRadius="5.%"   PorcentajeOPíxel
        BarGaugeMaxHeight="20.%"   PorcentajeOPíxel
        RoundGaugeNeedleLength="45.%"   PorcentajeOPíxel
        BarGaugeNeedleLength="3.%"   PorcentajeOPíxel
        >
        <TicksFont
            Color="#a0c0b0"
            Name="Tahoma"
            Bold="0"
            Italic="0"
            Underline="0"
            MinFontHeight="10.pt"
            Size="4.%"
        />
        <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>
```

Ejemplo: funciones para gráficos

A continuación puede ver un ejemplo de documento XSLT que muestra cómo usar las funciones de extensión para gráficos de Altova. Después ofrecemos un documento XML y una captura de

pantalla de la imagen generada cuando el documento XML se procesa con el documento XSLT usando el motor XSLT 2.0 o 3.0.

Nota: las funciones para gráficos solamente son compatibles con las ediciones **Enterprise Edition** y **Server** de los productos de Altova.

Nota: para más información sobre las tablas de datos del gráfico, consulte la documentación de las herramientas XMLSpy y StyleVision de Altova.

Documento XSLT

Este documento XSLT usa las funciones de extensión para gráficos de Altova para generar un gráfico circular. Este XSLT se puede usar para procesar el documento XML que aparece más abajo.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:altovaext="http://www.altova.com/xslt-extensions"
   exclude-result-prefixes="#all">
   <xsl:output version="4.0" method="html" indent="yes" encoding="UTF-8"/>
   <xsl:template match="/">
      <html>
         <head>
            <title>
               <xsl:text>HTML Page with Embedded Chart</xsl:text>
            </title>
         </head>
         <body>
            <xsl:for-each select="/Data/Region[1]">
               <xsl:variable name="extChartConfig" as="item()*">
                  <xsl:variable name="ext-chart-settings" as="item()*">
                     <chart-config>
                        <General
                           SettingsVersion="1"
                           ChartKind="Pie3d"
                           BKColor="#ffffff"
                           ShowBorder="1"
                           PlotBorderColor="#000000"
                           PlotBKColor="#ffffff"
                           Title="{@id}"
                           ShowLegend="1"
                           OutsideMargin="3.2%"
                           TitleToPlotMargin="3.%"
                           LegendToPlotMargin="6.%"
                           >
                           <TitleFont
                              Color="#023d7d"
                              Name="Tahoma"
                              Bold="1"
```

```
                                    Italic="0"
                                    Underline="0"
                                    MinFontHeight="10.pt"
                                    Size="8.%" />
                                </General>
                            </chart-config>
                        </xsl:variable>
                        <xsl:sequence select="altovaext:create-chart-config-from-
xml( $ext-chart-settings )"/>
                    </xsl:variable>
                    <xsl:variable name="chartDataSeries" as="item()*">
                        <xsl:variable name="chartDataRows" as="item()*">
                            <xsl:for-each select="(Year)">
                                <xsl:sequence select="altovaext:create-chart-data-
row( (@id), ( .) )"/>
                            </xsl:for-each>
                        </xsl:variable>
                        <xsl:variable name="chartDataSeriesNames" as="xs:string*"
select=" ( (&quot;Series 1&quot;), &apos;&apos; )[1]"/>
                        <xsl:sequence
                            select="altovaext:create-chart-data-series-from-
rows( $chartDataSeriesNames, $chartDataRows)"/>
                    </xsl:variable>
                    <xsl:variable name="ChartObj" select="altovaext:create-
chart( $extChartConfig, ( $chartDataSeries), false() )"/>
                    <xsl:variable name="sChartFileName" select="'mychart1.png'"/>
                    <img src="{$sChartFileName, altovaext:save-chart-
image( $ChartObj, $sChartFileName, 400, 400 ) }"/>
                </xsl:for-each>
            </body>
        </html>
    </xsl:template>
</xsl:stylesheet>
```

Documento XML

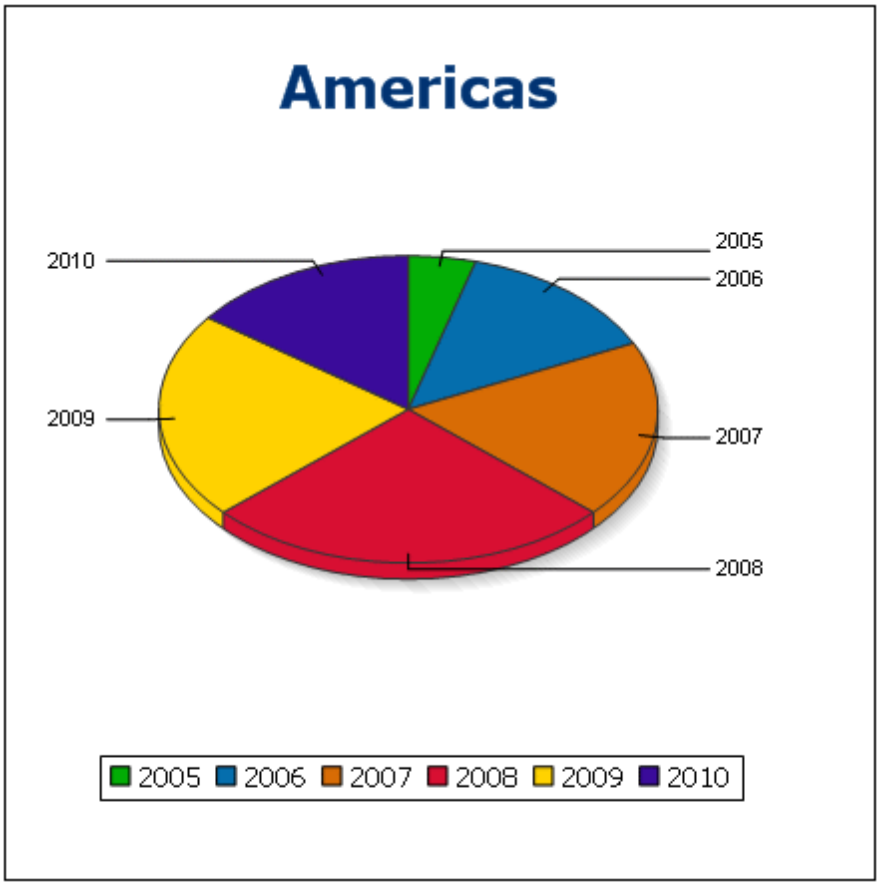
Este documento XML se puede procesar con el XSLT que aparece arriba. Los datos del documento XML se usan para generar el gráfico circular de la imagen que aparece más abajo.

```
<?xml version="1.0" encoding="UTF-8"?>
<Data xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="YearlySales.xsd">
   <ChartType>Pie Chart 2D</ChartType>
   <Region id="Americas">
       <Year id="2005">30000</Year>
       <Year id="2006">90000</Year>
       <Year id="2007">120000</Year>
       <Year id="2008">180000</Year>
       <Year id="2009">140000</Year>
       <Year id="2010">100000</Year>
   </Region>
```

```
    <Region id="Europe">
        <Year id="2005">50000</Year>
        <Year id="2006">60000</Year>
        <Year id="2007">80000</Year>
        <Year id="2008">100000</Year>
        <Year id="2009">95000</Year>
        <Year id="2010">80000</Year>
    </Region>
    <Region id="Asia">
        <Year id="2005">10000</Year>
        <Year id="2006">25000</Year>
        <Year id="2007">70000</Year>
        <Year id="2008">110000</Year>
        <Year id="2009">125000</Year>
        <Year id="2010">150000</Year>
    </Region>
</Data>
```

Imagen de salida

Este gráfico circular se generó al procesar el documento XML anterior con el archivo XSLT que aparece al principio de este apartado.



10.1.6 Funciones para códigos de barras

Los motores XSLT de Altova usan bibliotecas Java de terceros para crear códigos de barras. A continuación enumeramos las clases y los métodos públicos utilizados. Las clases se empaquetan en `AltovaBarcodeExtension.jar`, que está en la carpeta `<CarpetaArchivosPrograma>\Altova\Common2014\jar`.

Las bibliotecas Java utilizadas están en las subcarpetas de la carpeta `<CarpetaArchivosPrograma>\Altova\Common2014\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.

 © 2014 Altova GmbH

Se utilizan estas clases:

class **QRCodeBean**

- *Extiende* `org.krysalis.barcode4j.impl.AbstractBarcodeBean`
- *Crea una interfaz* `AbstractBarcodeBean` *para* `com.google.zxing.qrcode.encoder`

```
void generateBarcode(CanvasProvider canvasImp, String msg)
void setQRErrorCorrectionLevel(QRCodeErrorCorrectionLevel level)
BarcodeDimension calcDimensions(String msg)
double getVerticalQuietZone()
double getBarWidth()
```

class **QRCodeErrorCorrectionLevel** *Nivel de corrección de errores para* `QRCode`

```
static QRCodeErrorCorrectionLevel byName(String name)
"L" = ~7% correction
"M" = ~15% correction
"H" = ~25% correction
"Q" = ~30% correction
```

Ejemplo XSLT

A continuación puede ver un ejemplo de XSLT que ilustra el uso de funciones para códigos de barras en una hoja de estilos XSLT.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:fn="http://www.w3.org/2005/xpath-functions"
   xmlns:altova="http://www.altova.com"
   xmlns:altovaext="”http://www.altova.com/xslt-extensions”
   xmlns:altovaext-barcode="java:com.altova.extensions.barcode.BarcodeWrapper"
   xmlns:altovaext-barcode-
property="java:com.altova.extensions.barcode.BarcodePropertyWrapper">
   <xsl:output method="html" encoding="UTF-8" indent="yes"/>
   <xsl:template match="/">
      <html>
         <head><title/></head>
         <body>
            <img alt="barcode" src="{altovaext:get-temp-folder()}barcode.png"/>
         </body>
      </html>
      <xsl:result-document
         href="{altovaext:get-temp-folder()}barcode.png"
         method="text" encoding="base64tobinary" >
         <xsl:variable name="barcodeObject"
            select="altovaext-
barcode:newInstance(&apos;Code39&apos;,string(&apos;some value&apos;),
            96,0, (altovaext-barcode-property:new( &apos;setModuleWidth&apos;,
25.4 div 96 * 2 ) ) )"/>
         <xsl:value-of select="xs:base64Binary(xs:hexBinary(string(altovaext-
barcode:generateBarcodePngAsHexString($barcodeObject)) ))"/>
      </xsl:result-document>
```

```
    </xsl:template>
</xsl:stylesheet>
```


© 2014 Altova GmbH

10.2 Funciones de extensión varias

Los lenguajes de programación como Java y C# ofrecen varias funciones predefinidas que no están disponibles como funciones XQuery/XPath ni XSLT. Un ejemplo son las funciones matemáticas de Java `sin()` y `cos()`. Si los diseñadores de hojas de estilos XSLT y consultas XQuery tuvieran acceso a estas funciones, el área de aplicación de sus hojas de estilos y consultas aumentaría y su trabajo sería un poco más sencillo.

Los motores XSLT y XQuery de los productos de Altova admiten el uso de funciones de extensión en Java y .NET, así como scripts MSXSL para XSLT. También son compatibles con funciones XBRL para XSLT.

Esta sección describe cómo usar funciones de extensión y scripts MSXSL en hojas de estilos XSLT y documentos XQuery. Las funciones de extensión pueden organizarse en varios grupos:

- Funciones de extensión de Altova
- Funciones de extensión Java
- Funciones de extensión .NET
- Funciones XBRL para XSLT
- Scripts MSXSL para XSLT

En los apartados de esta sección nos ocupamos de tres aspectos fundamentales: (i) cómo se llaman las funciones en sus respectivas bibliotecas, (ii) qué reglas deben seguirse para convertir los argumentos de una llamada a función en el formato de entrada necesario de la función y (iii) qué reglas deben seguirse para la conversión del tipo devuelto.

Requisitos
Para que estas funciones de extensión funcionen es necesario tener Java Runtime Environment (para las funciones Java) y .NET Framework 2.0 o superior (para las funciones .NET) instalado en el equipo que ejecuta la transformación XSLT o XQuery.

10.2.1 Funciones de extensión Java

Puede usar una función de extensión Java dentro de una expresión XPath o XQuery para invocar un constructor Java o llamar a un método Java (estático o de instancia).

Un campo de una clase Java se trata como un método sin argumentos. Un campo puede ser estático o de instancia. Más adelante describimos cómo se accede a los campos estáticos y de instancia.

Este apartado tiene varias partes:

- Constructores Java
- Métodos estáticos y campos estáticos
- Métodos de instancia y campos de instancia
- Tipos de datos: conversión de XPath/XQuery en Java
- Tipos de datos: conversión de Java en XPath/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)" />
```

 © 2014 Altova GmbH

```
<xsl:value-of xmlns:jmath="java"
              select="jmath:java.lang.Math.cos(3.14)" />
```

El método nombrado en la función de extensión (`cos()`) debe coincidir con el nombre de un método estático público de la clase Java nombrada (`java.lang.Math`).

Ejemplo de código XQuery
Aquí puede ver un ejemplo de código XQuery similar al código XSLT anterior:

```
<cosine xmlns:jMath="java:java.lang.Math">
   {jMath:cos(3.14)}
</cosine>
```

Clases Java definidas por el usuario
Si creó sus propias clases Java, a los métodos de estas clases se les llama de otra manera, dependiendo de: (i) si a las clases se accede por medio de un archivo JAR o de un archivo de clases y (ii) si estos archivos están en el directorio actual (el directorio del documento XSLT o XQuery). Para más información consulte los apartados Archivos de clases definidos por el usuario 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.

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*.)
- 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*.)
- El archivo de clases está en un paquete. El archivo XSLT/XQuery está en una carpeta cualquiera. (*ver ejemplo*.)
- El archivo de clases no está en un paquete. El archivo XSLT/XQuery está una carpeta cualquiera. (*ver ejemplo*.)

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 `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>
```

 © 2014 Altova GmbH

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.

Archivos JAR definidos por el usuario

Si se accede a las clases por medio de un archivo JAR, entonces se debe especificar el URI del archivo JAR usando esta sintaxis:

```
xmlns:claseEspacioNombres="java:nombreClase?ruta=jar:uri-del-
archivoJar!/"
```

Para la llamada al método se usa el prefijo del URI de espacio de nombres que identifica la clase: `claseEspacioNombres:método()`

En la sintaxis anterior:

`java:` indica que se está llamando a una función de Java
`nombreClase` es el nombre de la clase definida por el usuario
`?` es el separador entre el nombre de la clase y la ruta de acceso
`ruta=jar:` indica que se ofrece una ruta de acceso a un archivo JAR
`uri-del-archivoJar` es el URI del archivo JAR
`!/` 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.

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" />
```

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>
```

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:

 © 2014 Altova GmbH

```
<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`.

Tipos de datos: conversión de XPath/XQuery en Java

Cuando se llama a una función Java desde dentro de una expresión XPath/XQuery, el tipo de datos de los argumentos de la función es importante a la hora de determinar a cuál de las clases Java que tienen el mismo nombre se llama.

En Java se siguen estas reglas:

- Si hay más de un método Java con el mismo nombre, pero cada método tiene un número diferente de argumentos, entonces se selecciona el método Java que mejor se ajusta al número de argumentos de la llamada a función.
- Los tipos de datos de cadena, numéricos y booleanos de XPath/XQuery (*ver lista más abajo*) se convierten de forma implícita en el tipo de datos Java correspondiente. Si el tipo XPath/XQuery suministrado se puede convertir a más de un tipo Java (p. ej. `xs:integer`), entonces se selecciona el tipo Java que se declaró para el método seleccionado. Por ejemplo, si el método Java al que se llama es `fx(decimal)` y el tipo de datos XPath/XQuery suministrado es `xs:integer`, entonces `xs:integer` se convierte en el tipo de datos Java `decimal`.

La tabla que aparece a continuación enumera las conversiones implícitas de los tipos de cadena, numéricos y booleanos XPath/XQuery en tipos de datos Java.

`xs:string`	`java.lang.String`

xs:boolean	boolean (primitivo), java.lang.Boolean
xs:integer	int, long, short, byte, float, double y sus clases contenedoras, como java.lang.Integer
xs:float	float (primitivo), java.lang.Float, double (primitivo)
xs:double	double (primitivo), java.lang.Double
xs:decimal	float (primitivo), java.lang.Float, double(primitivo), java.lang.Double

Los subtipos de los tipos de datos XML Schema de la tabla anterior (que se usan en XPath y XQuery) también se convierten en los tipos Java correspondientes al tipo antecesor del subtipo.

En algunos casos quizás no sea posible seleccionar el método Java correcto usando la información dada. Por ejemplo, imagine que:

- El argumento presentado es un valor xs:untypedAtomic de 10 y está destinado al método mimétodo(float).
- Sin embargo, hay otro método en la clase que toma un argumento de otro tipo de datos: mimétodo(double).
- Puesto que los métodos tienen el mismo nombre y el tipo suministrado (xs:untypedAtomic) se puede convertir correctamente tanto en float como en double, es posible que xs:untypedAtomic se convierta en double en lugar de en float.
- Por consiguiente, el método seleccionado no será el método necesario y quizás no produzca el resultado esperado. Una solución es crear un método definido por el usuario con un nombre diferente y usar ese método.

Los tipos que no aparecen en la lista anterior (p. ej. xs:date) no se convertirán y generarán un error. No obstante, tenga en cuenta que en algunos casos, es posible crear el tipo Java necesario usando un constructor Java.

Tipos de datos: conversión de Java en XPath/XQuery

Cuando un método Java devuelve un valor y el tipo de datos del valor es un tipo de cadena, numérico o booleano, entonces se convierte en el tipo de datos XPath/XQuery correspondiente. Por ejemplo, los tipos de datos Java java.lang.Boolean y boolean se convierten en xsd:boolean.

Las matrices unidimensionales devueltas por las funciones se extienden en una secuencia. Las matrices multidimensionales no se convierten y, por tanto, deberían ser contenidas.

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).

 © 2014 Altova GmbH

10.2.2 Funciones de extensión .NET

Si trabaja en la plataforma .NET desde un equipo Windows, puede usar funciones de extensión escritas en cualquier lenguaje .NET (p. ej. C#). Una función de extensión .NET se puede usar dentro de una expresión XPath/XQuery para invocar un constructor, una propiedad o un método (estático o de instancia) de una clase .NET.

A una propiedad de una clase .NET se le llama usando la sintaxis `get_NombrePropiedad()`.

Este apartado tiene varias partes:

- Constructores
- Métodos estáticos y campos estáticos
- Métodos de instancia y campos de instancia
- Tipos de datos: conversión de XPath/XQuery en .NET
- Tipos de datos: conversión de .NET en XPath/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" />
```

 © 2014 Altova GmbH

```
   <xsl:template match="/">
      <math xmlns:math="clitype:System.Math">
         <sqrt><xsl:value-of select="math:Sqrt(9)"/></sqrt>
         <pi><xsl:value-of select="math:PI()"/></pi>
         <e><xsl:value-of select="math:E()"/></e>
         <pow><xsl:value-of select="math:Pow(math:PI(), math:E())"/></pow>
      </math>
   </xsl:template>
</xsl:stylesheet>
```

La declaración de espacio de nombres del elemento `math` asocia el prefijo `math:` al URI `clitype:System.Math`. La parte inicial `clitype:` del URI indica que lo que sigue identifica una clase del sistema o una clase cargada. El prefijo `math:` de las expresiones XPath asocia las funciones de extensión al URI (y, por extensión, a la clase) `System.Math`. Las funciones de extensión identifican métodos en la clase `System.Math` y presenta argumentos cuando es necesario.

Ejemplo de código XQuery
Aquí puede ver un fragmento de código XQuery similar al ejemplo anterior:

```
<math xmlns:math="clitype:System.Math">
   {math:Sqrt(9)}
</math>
```

Tal y como ocurre con el código XSLT anterior, la declaración de espacio de nombres identifica la clase .NET, en este caso una clase del sistema. La expresión XQuery identifica el método al que se debe llamar y presenta el argumento.

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" />
```

Metodos estáticos y campos estáticos

La llamada a un método estático la hace directamente su nombre y se hace presentando los argumentos para el método. El nombre usado en la llamada debe ser el mismo que un método estático público de la clase especificada. Si el nombre del método y el número de argumentos que se dio en la llamada a función coincide con algún método de la clase, entonces los tipos de los argumentos presentados se evalúan para encontrar el resultado ideal. Si no se encuentra ninguna coincidencia, se emite un error.

Nota: un campo de una clase .NET se trata como si fuera un método sin argumentos. Para llamar a una propiedad se usa la sintaxis `get_nombrePropiedad()`.

Ejemplos

Este ejemplo de código XSLT muestra una llamada a un método con un argumento (`System.Math.Sin(arg)`):

```
<xsl:value-of select="math:Sin(30)" xmlns:math="clitype:System.Math"/>
```

Este ejemplo de código XSLT muestra una llamada a un campo (que se trata como si fuera un método sin argumentos) (`System.Double.MaxValue()`):

```
<xsl:value-of select="double:MaxValue()"
xmlns:double="clitype:System.Double"/>
```

Este ejemplo de código XSLT muestra una llamada a una propiedad (la sintaxis es `get_nombrePropiedad()`) (`System.String()`):

 © 2014 Altova GmbH

```
<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>
```

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.

Tipos de datos: conversión de XPath/XQuery en .NET

Cuando se usa una función de extensión .NET dentro de una expresión XPath/XQuery, los tipos de datos de los argumentos de la función son importantes para determinar a cuál de los métodos .NET que tienen el mismo nombre se está llamando.

En .NET se siguen estas normas:

- Si en una clase hay varios métodos que tienen el mismo nombre, solamente se pueden seleccionar los métodos que tienen el mismo número de argumentos que la llamada a función.
- Los tipos de datos de cadena, numéricos y booleanos XPath/XQuery (*ver lista más abajo*) se convierten de forma implícita en el tipo de datos .NET correspondiente. Si el tipo XPath/XQuery suministrado se puede convertir en más de un tipo .NET (p. ej. `xs:integer`), entonces se selecciona el tipo .NET que se declaró para el método seleccionado. Por ejemplo, si el método .NET al que se está llamando es `fx(double)` y el tipo de datos XPath/XQuery suministrado es `xs:integer`, entonces se convierte `xs:integer` en el tipo de datos .NET `double`.

La tabla que aparece a continuación enumera las conversiones implícitas de los tipos de cadena, numéricos y booleanos XPath/XQuery en tipos de datos .NET.

xs:string	StringValue, string
xs:boolean	BooleanValue, bool
xs:integer	IntegerValue, decimal, long, integer, short, byte, double, float
xs:float	FloatValue, float, double
xs:double	DoubleValue, double
xs:decimal	DecimalValue, decimal, double, float

Los subtipos de los tipos de datos XML Schema de la tabla anterior (que se usan en XPath y XQuery) también se convierten en los tipos .NET correspondientes al tipo antecesor del subtipo.

En algunos casos quizás no sea posible seleccionar el método .NET correcto usando la información dada. Por ejemplo, imagine que:

- El argumento presentado es un valor `xs:untypedAtomic` de 10 y está destinado al método `mimétodo(float)`.
- Sin embargo, hay otro método en la clase que toma un argumento de otro tipo de datos: `mimétodo(double)`.
- Puesto que los métodos tienen el mismo nombre y el tipo suministrado (`xs:untypedAtomic`) se puede convertir correctamente tanto en `float` como en `double`, es posible que `xs:untypedAtomic` se convierta en `double` en lugar de en `float`.
- 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.

Tipos de datos: conversión de .NET en XPath/XQuery

Cuando un método .NET devuelve un valor y el tipo de datos del valor es un tipo de cadena, numérico o booleano, entonces se convierte en el tipo de datos XPath/XQuery correspondiente. Por ejemplo, el tipo de datos .NET `decimal` se convierte en `xsd:decimal`.

Cuando se devuelve un objeto .NET o un tipo de datos que no es de cadena, numérico ni booleano, puede garantizar la conversión del tipo XPath/XQuery necesario usando primero un método .NET (p. ej. `System.DateTime.ToString()`) para convertir el objeto .NET en una cadena. En XPath/XQuery la cadena se puede modificar para ajustarse a la representación léxica del tipo necesario y convertirse después en dicho tipo (usando la expresión `cast as`, por ejemplo).

10.2.3 Funciones XBRL para XSLT

Puede llamar desde un contexto XSLT a todas las funciones definidas en el registro de funciones XBRL para transformar documentos de instancia XBRL. Estas funciones XBRL se definen en uno de estos dos espacios de nombres:

`http://www.xbrl.org/2008/function/instance` (que suele utilizarse con el prefijo `xfi:`)
`http://www.xbrl.org/2010/function/formula` (que suele utilizarse con el prefijo `xff:`)

Por ejemplo, la función XBRL `xfi:context` se amplía hasta `http://www.xbrl.org/2008/function/instance:context` (asumiendo que este espacio de nombres se enlazara al prefijo `xfi:`).

Para ver la lista de funciones visite el sitio http://www.xbrl.org/functionregistry/functionregistry.xml.

 © 2014 Altova GmbH

10.2.4 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" />

   ...

</msxsl:script>
```

El valor del atributo `namespace` es el nombre del espacio de nombres.

Altova RaptorXML+XBRL Server 2014

Altova LicenseServer

11 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, Linux o Mac OS X).
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: la página de configuración de LicenseServer no es compatible con SSL.

Versiones de LicenseServer y compatibilidad con los productos servidor de Altova
Cuando actualice su producto servidor de Altova a una versión más reciente, asegúrese de actualizar también su servidor LicenseServer con la versión más reciente. Las versiones anteriores de los productos servidor de Altova funcionarán con las conversiones más recientes de LicenseServer.

Es decir, si instala una versión nueva de un producto servidor de Altova y la versión actual de su servidor LicenseServer no es la más reciente, desinstale la versión antigua de LicenseServer e instale la versión más reciente (disponible en el sitio web de Altova). Toda la información de registro y licencias almacenada en la versión antigua se guardará en una base de datos su equipo servidor durante la desinstalación y se importará automáticamente a la versión nueva.

Versión actual: 1.8

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

 © 2014 Altova GmbH

Altova LicenseServer.

- Referencia de la página de configuración: describe la interfaz de administración de LicenseServer y sus funciones.

Última actualización: 23/04/2014

11.1 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.

 © 2014 Altova GmbH

11.2 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.

Para aprender a asignar licencias con Altova LicenseServer consulte la sección Asignación de licencias.

Versiones de LicenseServer y compatibilidad con los productos servidor de Altova
Cuando actualice su producto servidor de Altova a una versión más reciente, asegúrese de actualizar también su servidor LicenseServer con la versión más reciente. Las versiones anteriores de los productos servidor de Altova funcionarán con las conversiones más recientes de LicenseServer.

Es decir, si instala una versión nueva de un producto servidor de Altova y la versión actual de su servidor LicenseServer no es la más reciente, desinstale la versión antigua de LicenseServer e instale la versión más reciente (disponible en el sitio web de Altova). Toda la información de registro y licencias almacenada en la versión antigua se guardará en una base de datos su equipo servidor durante la desinstalación y se importará automáticamente a la versión nueva.

Versión actual: 1.8

El número de versión de su servidor LicenseServer actual aparece en la parte inferior de la página de configuración. El número de versión de LicenseServer también aparece durante la instalación (tanto de LicenseServer como del producto servidor que contiene el paquete de instalación de LicenseServer). Recuerde que puede instalar LicenseServer junto con el producto servidor o por separado. Si quiere actualizar LicenseServer con una versión más reciente, recuerde que antes debe desinstalar la versión anterior.

11.3 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-1.8-debian.deb
[Ubuntu]:   sudo dpkg --install licenseserver-1.8-ubuntu.deb
[CentOS, RedHat]:   sudo rpm -ivh licenseserver-1.8-1.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

 © 2014 Altova GmbH

licencias.

Versiones de LicenseServer y compatibilidad con los productos servidor de Altova
Cuando actualice su producto servidor de Altova a una versión más reciente, asegúrese de actualizar también su servidor LicenseServer con la versión más reciente. Las versiones anteriores de los productos servidor de Altova funcionarán con las conversiones más recientes de LicenseServer.

Es decir, si instala una versión nueva de un producto servidor de Altova y la versión actual de su servidor LicenseServer no es la más reciente, desinstale la versión antigua de LicenseServer e instale la versión más reciente (disponible en el sitio web de Altova). Toda la información de registro y licencias almacenada en la versión antigua se guardará en una base de datos su equipo servidor durante la desinstalación y se importará automáticamente a la versión nueva.

Versión actual: 1.8

11.4 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.LicenseServer.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/LicenseServer
```

Si necesita desinstalar una versión antigua de Altova LicenseServer, utilice el procedimiento descrito para FlowForce Server.

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

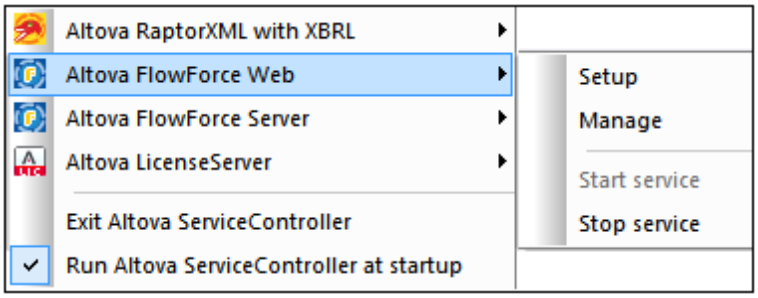
 © 2014 Altova GmbH

11.5 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.

11.6 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.

 © 2014 Altova GmbH

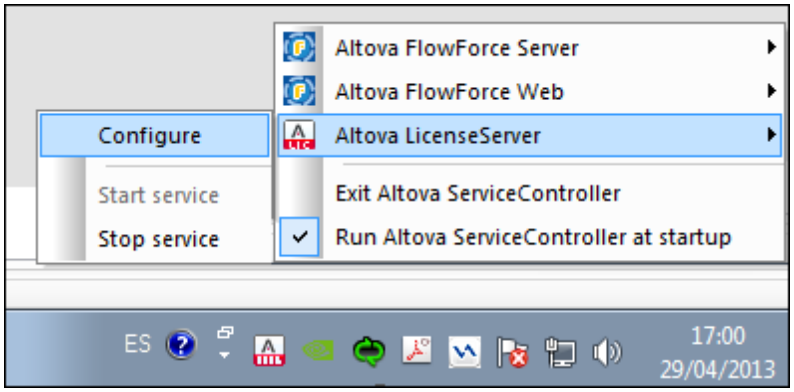
11.6.1 Iniciar LicenseServer

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.

Primero 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 (*imagen siguiente*). Si selecciona la opción **Run Altova ServiceController at Startup**, cuando inicie sesión en el sistema Altova ServiceController se iniciará 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 anterior*) y seleccione **Start Service** para iniciar el servicio. Si LicenseServer ya está en ejecución, la opción **Start Service** aparece deshabilitada (*imagen anterior*).

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.LicenseServer.plist
```

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

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

 © 2014 Altova GmbH

11.6.2 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

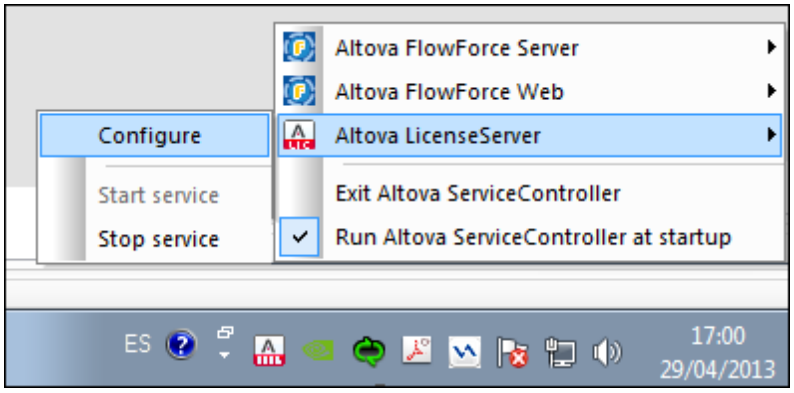
---

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 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:

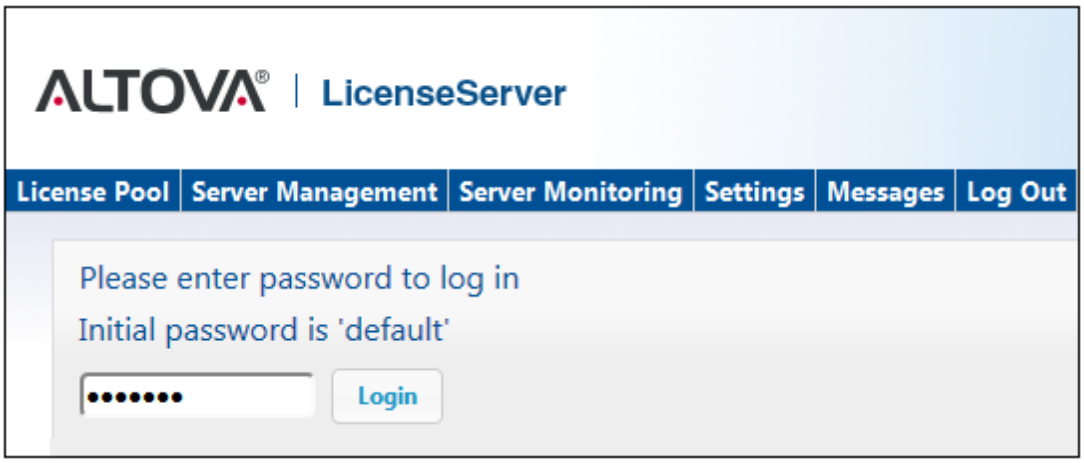
C:/ProgramData/Altova/LicenseServer/WebUI.html

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

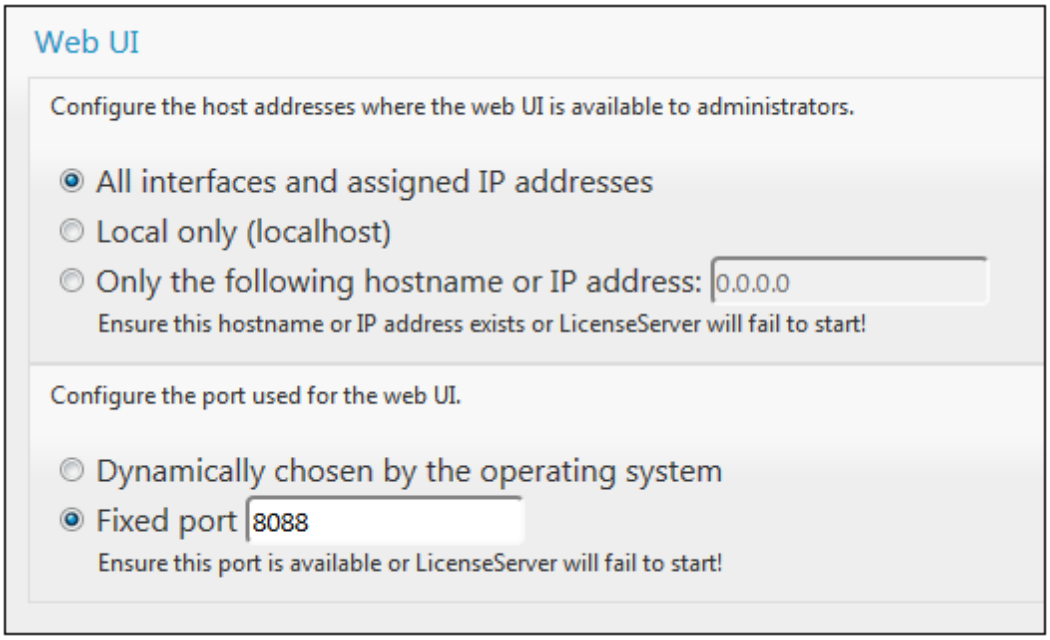
Después de seguir todas estas instrucciones se abre la página de configuración por la página de inicio de sesión (*imagen siguiente*). Puede iniciar sesión con la contraseña predeterminada `default`. Después de iniciar sesión puede cambiar la contraseña en la pestaña **Settings**.



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)).

 © 2014 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.

11.6.3 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:

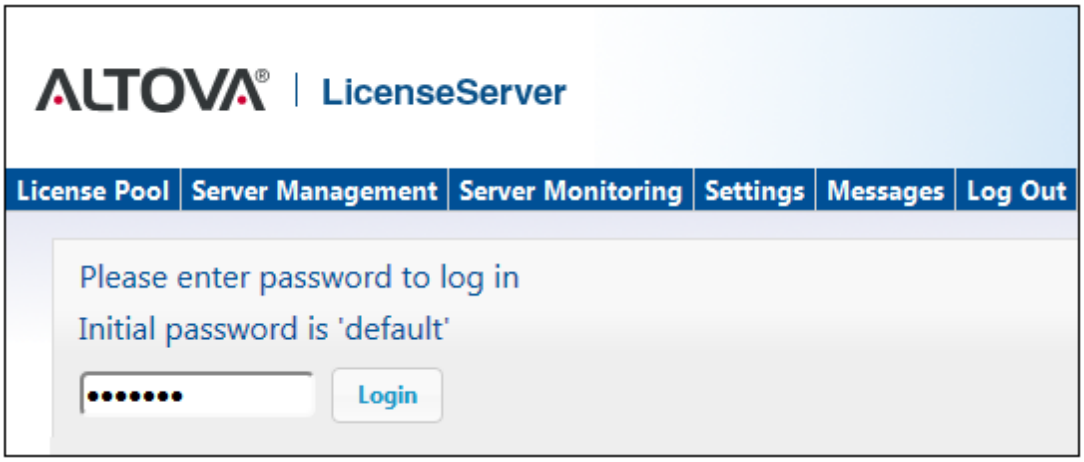
```
/var/opt/Altova/LicenseServer/webUI.html
```

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`**.

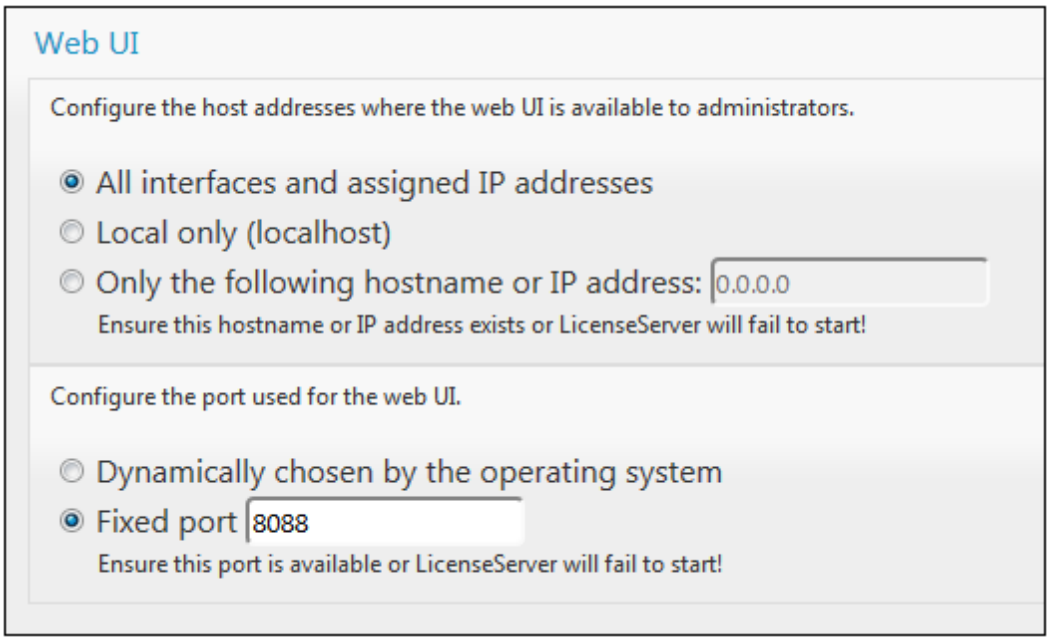
El formato de la URL generada dinámicamente en **`WebUI.html`** será similar a `http://127.0.0.1:55541` y se encuentra dentro de la función `checkIfServiceRunning()` en un script situado cerca del final del elemento `<head>`. El número de puerto de la URL se asigna dinámicamente y la parte de la dirección 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**.

 © 2014 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.

11.6.4 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
```

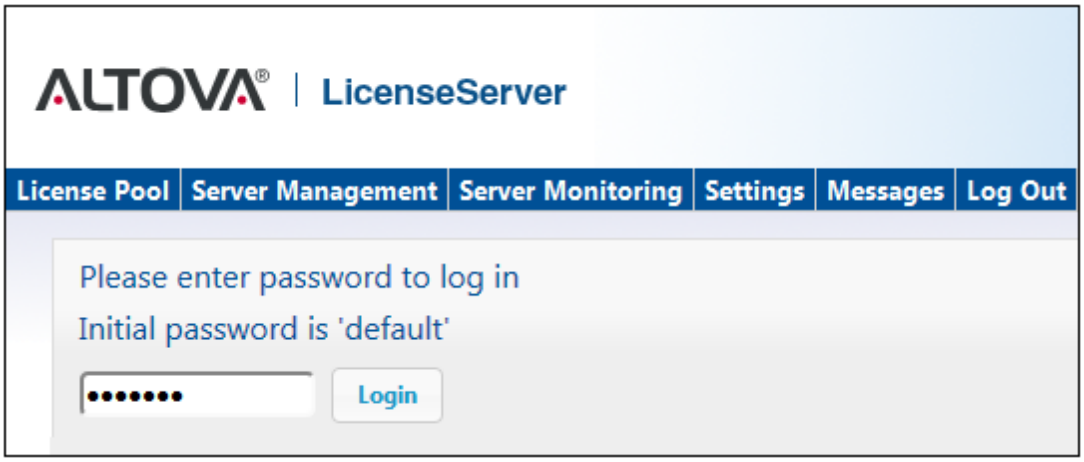
Si eligió que la URL de la página de configuración se genere de forma dinámica (en la pestaña **Configuración** de la página de configuración), se genera una URL nueva cada vez que se inicia LicenseServer. Para averiguar la URL actual de la página de configuración necesitará mirar la versión actual de **`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`.

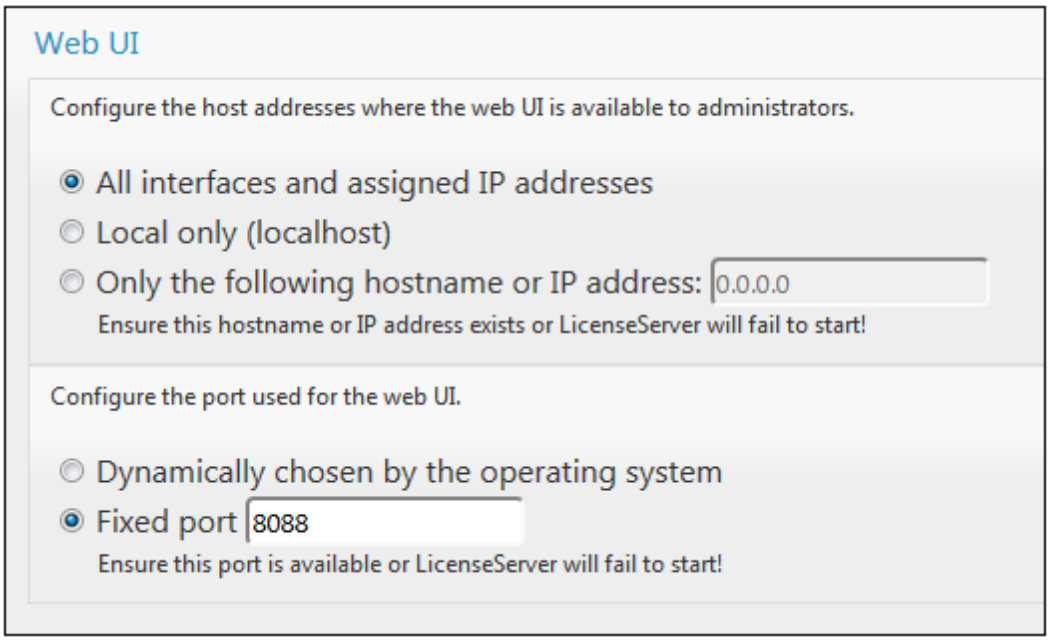
Nota: la página de configuración también se puede abrir directamente desde el icono **Finder | Aplicaciones | Altova License Server**.

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**.

 © 2014 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.

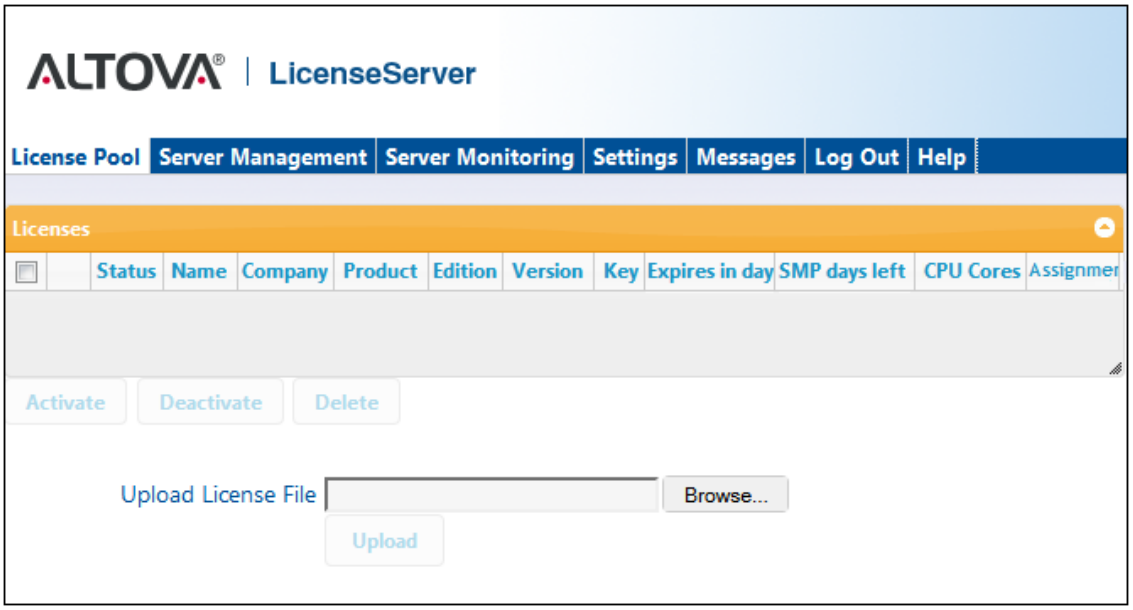
11.6.5 Cargar licencias en LicenseServer

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

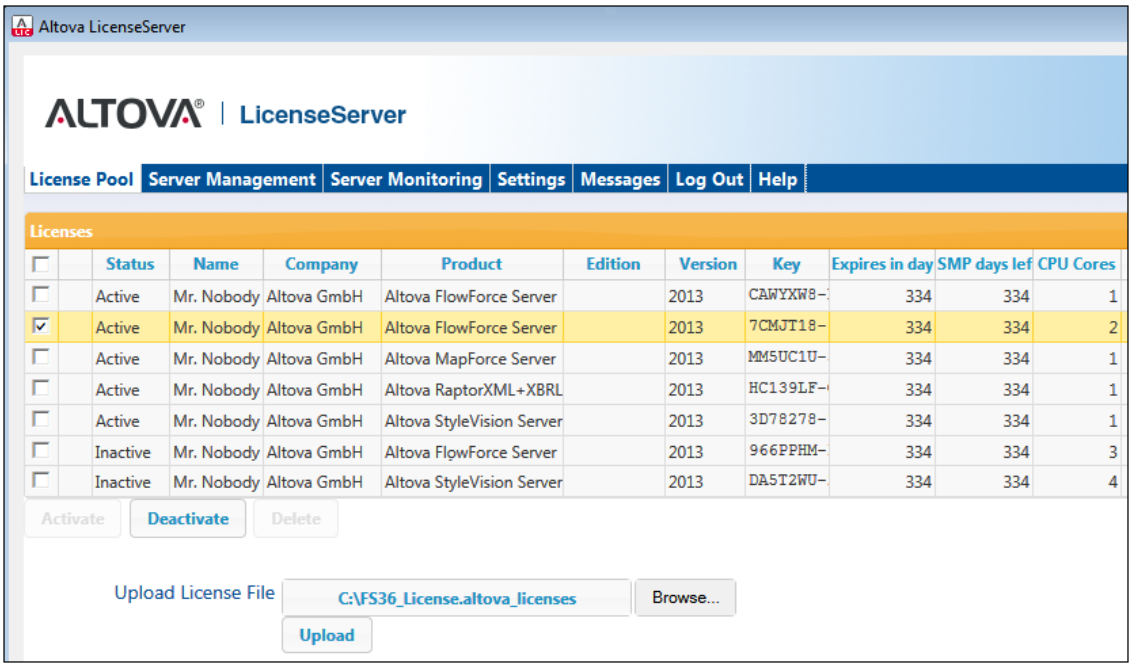
Cargar un archivo de licencias al repertorio de licencias de LicenseServer
Cuando reciba el archivo de licencias de Altova, cárguelo al repertorio de licencias de Altova LicenseServer (*ver instrucciones más abajo*). Un archivo de licencias puede contener varias licencias y al cargarlo, se cargan todas 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.

 © 2014 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.

 © 2014 Altova GmbH

11.6.6 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 2014 | 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:

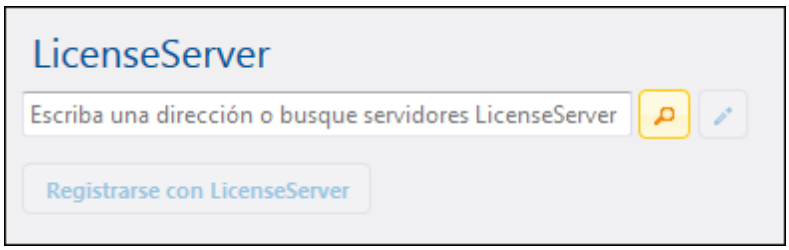
FlowForceWeb running on `http://127.0.1.1:3459/setup?key=52239315203`

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*).

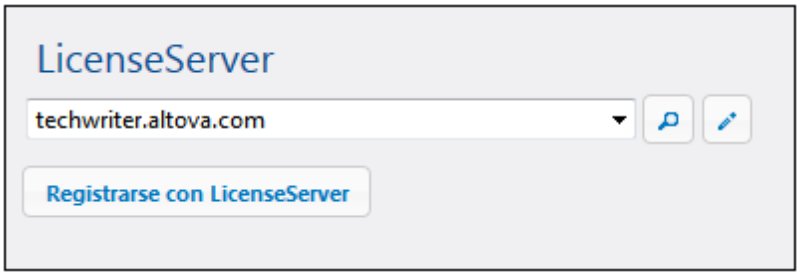
Registrar FlowForce Server desde la página de configuración (Windows/Linux)
En la página de configuración (*imagen siguiente*) en el campo *LicenseServer* se debe indicar el servidor Altova LicenseServer con el que desea registrar el producto.

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*).



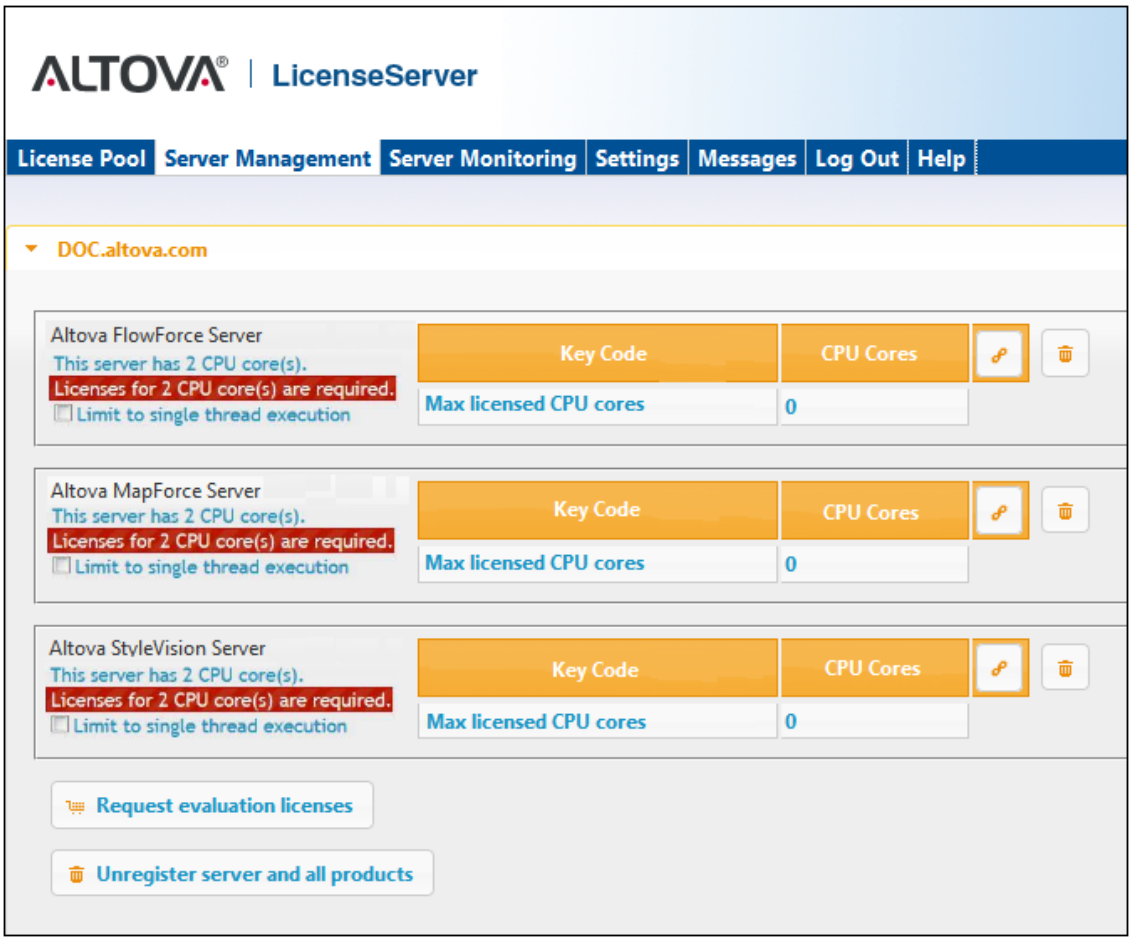
La búsqueda devuelve una lista con los servidores LicenseServer que están ejecutándose en la red. Seleccione el LicenseServer que almacena su licencia de FlowForce Server (*imagen siguiente*).



- 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/FlowForceServer2014/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/MapForceServer2014/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.

 © 2014 Altova GmbH

11.6.7 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/MapForceServer2014/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/MapForceServer2014/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.

 © 2014 Altova GmbH

11.6.8 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/StyleVisionServer2014/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/StyleVisionServer2014/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.

 © 2014 Altova GmbH

11.6.9 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 o en la misma red a la que está conectado LicenseServer y después debe 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 Server:	**`RaptorXML licenseserver`** ***`Servidor-O-Dirección-IP`***
RaptorXML+XBRL Server:	**`RaptorXMLXBRL licenseserver`** ***`Servidor-O-Dirección-IP`***

Por ejemplo, si LicenseServer se está ejecutando en `http://localhost:8088`, entonces debe registrar RaptorXML(+XBRL) Server con:

RaptorXML Server:	**`RaptorXML licenseserver hostlocal`**
RaptorXML+XBRL Server:	**`RaptorXMLXBRL licenseserver hostlocal`**

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/RaptorXMLServer2014/bin/raptorxmlserver licenseserver
localhost
sudo /opt/Altova/RaptorXMLXBRLServer2014/bin/raptorxmlxbrlserver 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/RaptorXMLServer2014/bin
/opt/Altova/RaptorXMLXBRLServer2014/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.

 © 2014 Altova GmbH

11.6.10 Asignar licencias a productos registrados

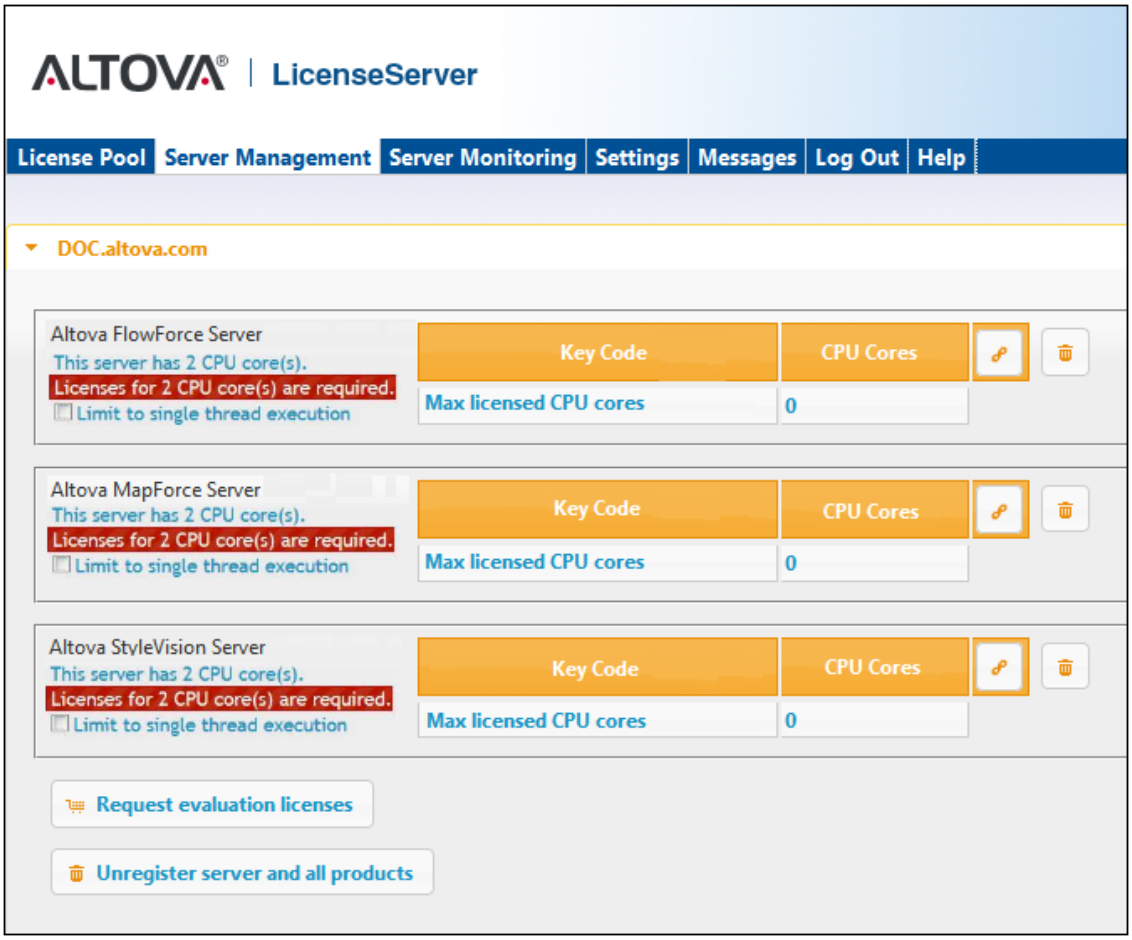
Este apartado explica:

- 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.
- si se instalaron **varias versiones** del mismo producto en el mismo equipo (p. ej. StyleVision Server 2013 y StyleVision Server 2014) y todas ellas se registraron con el mismo servidor LicenseServer, todos los registros se reúnen en un solo registro en la pestaña **Server Management**, es decir, aparecen como un solo registro. Cuando se asigna una licencia a este registro, esta se asigna a todas las instalaciones reunidas en ese registro. Sin embargo, en el equipo cliente se pueden ejecutar simultáneamente

varias instancias de una instalación solamente. Por ejemplo, puede ejecutar simultáneamente varias instancias de StyleVision Server 2013 o varias instancias de StyleVision Server 2014, pero puede ejecutar simultáneamente una instancia de StyleVision Server 2013 y otra de StyleVision Server 2014. Recuerde que si instala versiones nuevas, debe asignarles una licencia para poder ejecutarlas.

- cuando actualice su producto servidor de Altova a una versión más reciente, asegúrese de actualizar también su servidor LicenseServer con la versión más reciente. Las versiones anteriores de los productos servidor de Altova funcionarán con las conversiones más recientes de LicenseServer. Es decir, si instala una versión nueva de un producto servidor de Altova y la versión actual de su servidor LicenseServer no es la más reciente, desinstale la versión antigua de LicenseServer e instale la versión más reciente (disponible en el sitio web de Altova). Toda la información de registro y licencias almacenada en la versión antigua se guardará en una base de datos su equipo servidor durante la desinstalación y se importará automáticamente a la versión nueva. El número de versión de su servidor LicenseServer actual aparece en la parte inferior de la página de configuración. El número de versión de LicenseServer también aparece durante la instalación (tanto de LicenseServer como del producto servidor que contiene el paquete de instalación de LicenseServer). Recuerde que puede instalar LicenseServer junto con el producto servidor o por separado. Si quiere actualizar LicenseServer con una versión más reciente, recuerde que antes debe desinstalar la versión anterior.

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.

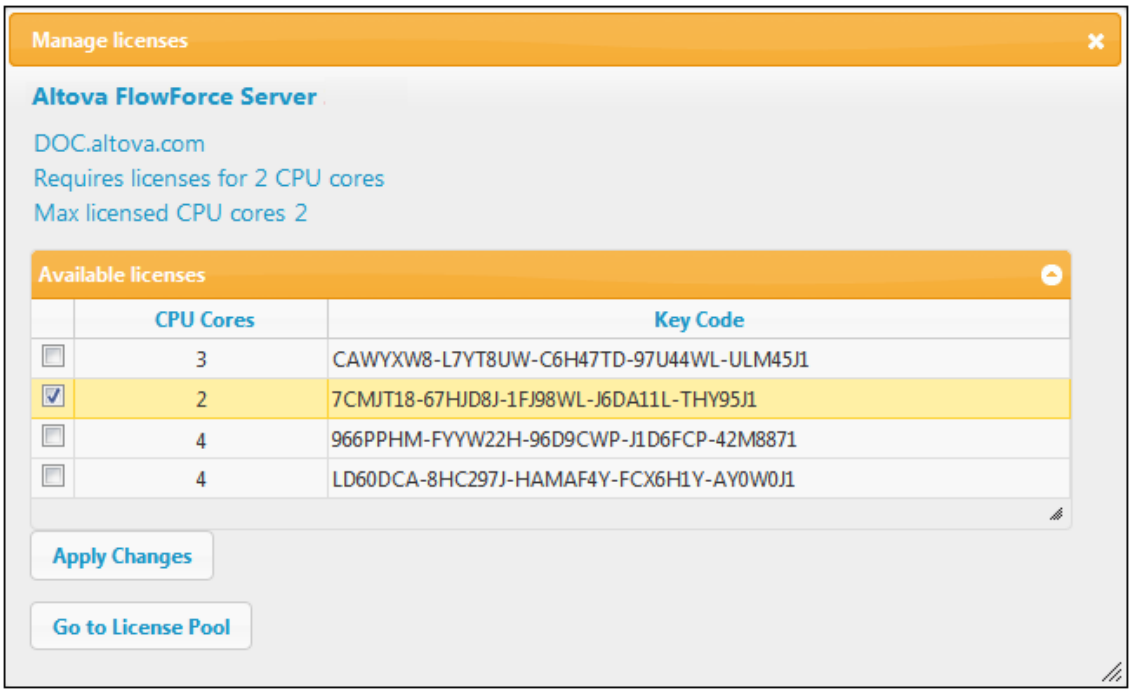
Por ejemplo, si un servidor tiene ocho núcleos (un procesador *octa-core*), deberá comprar una licencia para ocho núcleos. También puede combinar varias licencias para alcanzar el número de núcleos necesario. Es decir, puede usar dos licencias para cuatro núcleos para un servidor *octa-core* en lugar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



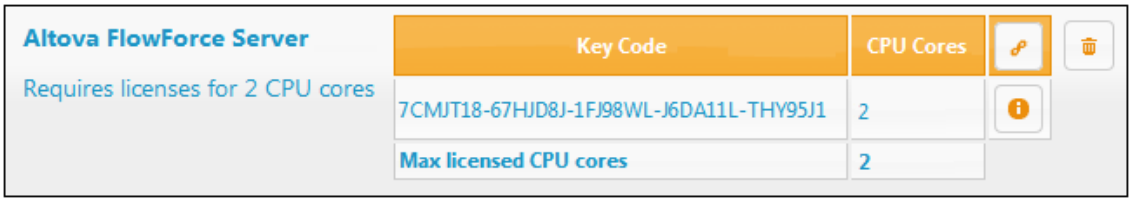
En este cuadro de diálogo:

- 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 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.

11.7 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, StyleVision Server, RaptorXML(+XBRL) 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.

 © 2014 Altova GmbH

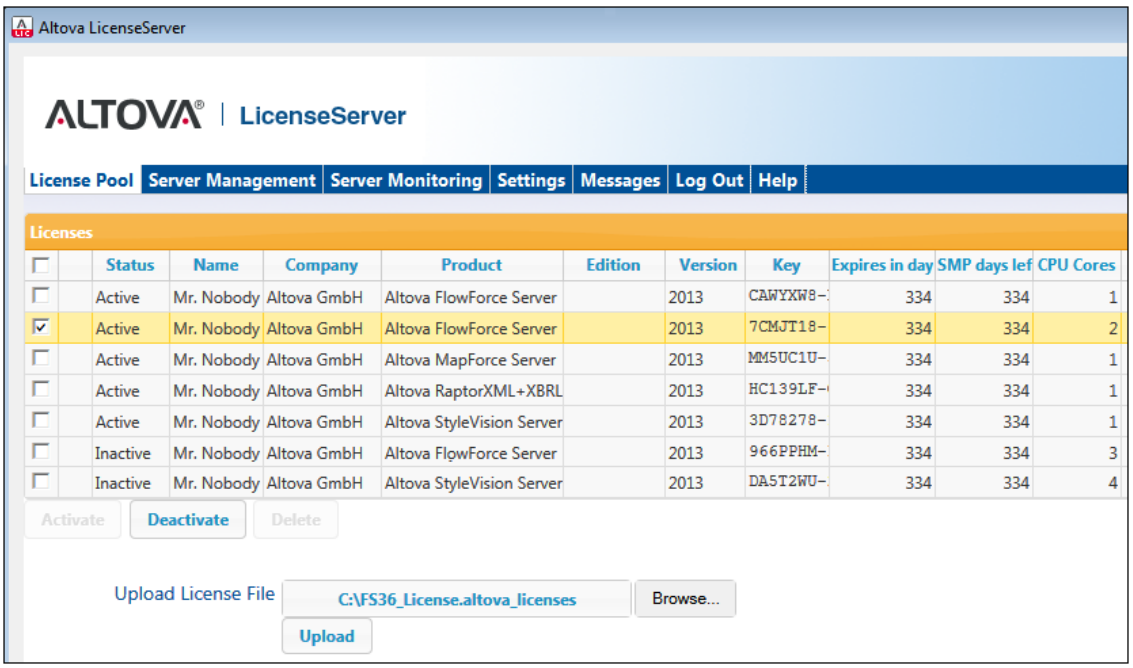
11.7.1 Repertorio de licencias

Temas de este apartado:

- Cargar una licencia
- Estado de las licencias
- Activar, desactivar y eliminar licencias
- Iconos de la pestaña **License Pool**
- Información sobre las licencias
- Nota sobre núcleos y 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.

 © 2014 Altova GmbH

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.

Iconos de la pestaña License Pool

Editar licencias asignadas: este icono aparece en cada licencia (en la columna *Assignments*). Abre el cuadro de diálogo de gestión de licencias, donde puede asignar

nuevas licencias al producto y editar las licencias asignadas previamente.

Mostrar licencias: este icono aparece en cada licencia (en la columna *Assignments*). Ofrece información sobre los clientes activos que están activos en ese momento.

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.
- ***Assignments***. Ofrece acceso a cuadros de diálogo de edición y a información sobre cada licencia.

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.

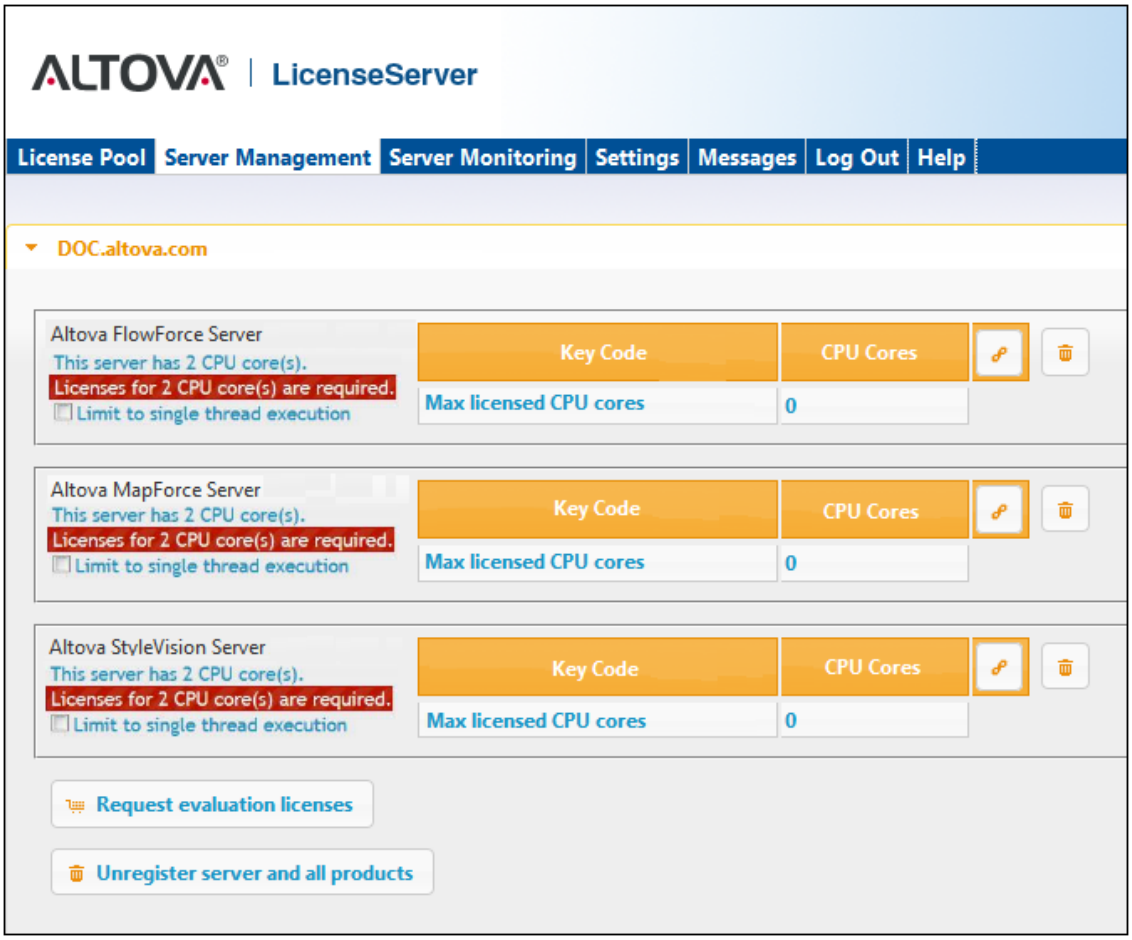
 © 2014 Altova GmbH

11.7.2 Gestión de servidores

Temas de este apartado:

- Iconos de la pestaña **Server Management**
- Asignación de licencias
- Un equipo cliente que usa nombres diferentes
- Solicitar una licencia de evaluación
- Quitar productos del registro

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.

Ejecución por subprocesamiento simple
Si en el repertorio de licencias hay una licencia de producto para un solo núcleo, puede asignar un equipo con varios núcleos a dicha licencia. En casos como este, el equipo ejecutará el producto en un solo núcleo. Por tanto, el procesamiento será más lento porque el subprocesamiento múltiple no estará disponible (funciona con varios núcleos solamente). El producto se ejecutará en modo de subprocesamiento simple en el equipo.

Para asignar una licencia para un solo núcleo a un equipo con varios núcleos, marque la casilla *Limit to single thread execution* del producto correspondiente.

Iconos de la pestaña Server Management

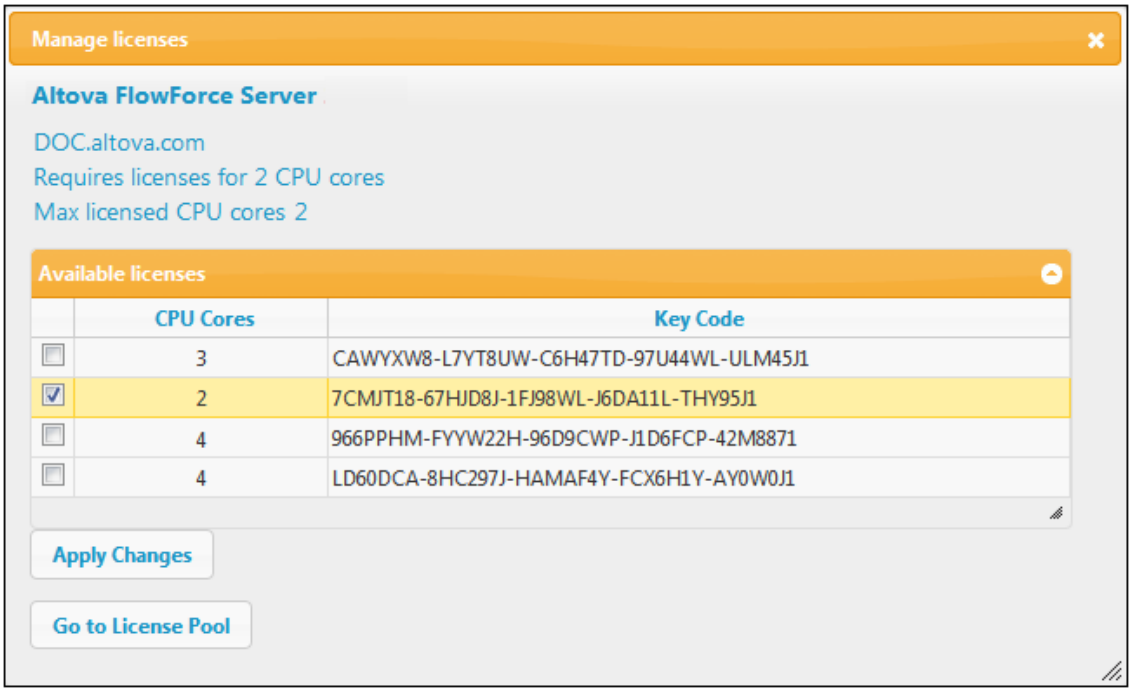
Editar licencias asignadas: abre el cuadro de diálogo de gestión de licencias, donde puede asignar nuevas licencias al producto y editar las licencias asignadas previamente.

Mostrar licencias: aparece con cada licencia. Abre la pestaña del repertorio de licencias y resalta la licencia seleccionada para que pueda ver sus detalles.

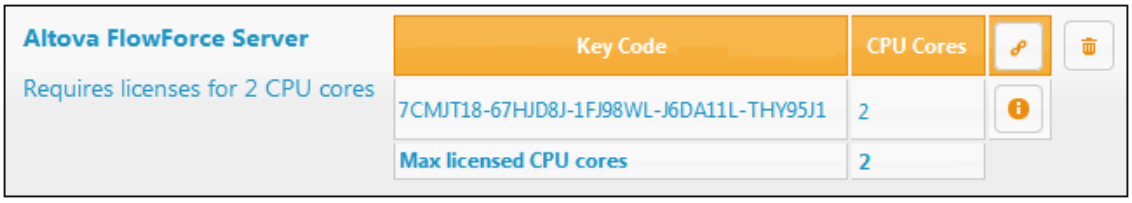
Quitar producto del registro: elimina el producto seleccionado (del equipo cliente seleccionado) del registro de LicenseServer.

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*).

 © 2014 Altova GmbH



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*).



Un equipo cliente que usa nombres diferentes
Si un equipo cliente se registra más de una vez con LicenseServer, puede aparecer en la pestaña **Server Management** con varios nombres diferentes, es decir, con varias entradas. Esto puede pasar si, por ejemplo, un equipo se vuelve a registrar con un nombre de host diferente.

Es importante que no se asignen licencias adicionales al mismo equipo bajo un nombre diferente. Por eso debe quitar la asignación de licencias a las instancias del equipo cliente que son redundantes (clic en el botón **Unregister server and all products** de los equipos redundantes).

Nota: aunque en esta documentación nos referimos a los equipos cliente como clientes de LicenseServer, en realidad son servidores de sus propios productos.

Además si se asigna varias veces la misma licencia al mismo equipo (pero con nombres diferentes), pueden producirse conflictos de licencia. Con lo cual, para evitar estas dos situaciones (asignación redundante de licencias y asignación de la misma licencia varias veces), recomendamos quitar del registro las entradas redundantes del mismo equipo cliente.

A continuación puede ver los nombres que puede tomar un equipo en la pestaña **Server**

Management:

- *Nombre de host con nombre del dominio (el nombre de dominio completo)*, por ejemplo "`win80-x64_1.mi.dominio.com`" o "`Doc3.mi.dominio.com`". Esto ocurre cuando el nombre de host del equipo (con o sin la información de dominio) se pasa como argumento del comando `licenseserver` de la línea de comandos que sirve para registrar el producto servidor con LicenseServer. Por ejemplo: `<ProductoServidorAltova> licenseserver Doc3`. Esto genera el nombre de dominio completo `Doc3.mi.dominio.com`.

  También se obtiene un nombre de dominio completo cuando en Windows 7 se aporta `localhost` como nombre de host.

- *Nombre de host sin nombre de dominio*, por ejemplo "`win80-x64_1`" o "`Doc3`". Esto ocurre en sistemas Windows 8 cuando se aporta `localhost` como nombre del equipo.

- *localhost*. En algunos casos también aparece `localhost` como nombre del equipo.

Nota: si durante la instalación del producto servidor de Altova en equipos Windows, el equipo se registra automáticamente con LicenseServer, el instalador usa `localhost` como nombre del equipo.

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 con LicenseServer, abra su página de configuración o su interfaz de la línea de comandos y regístrelo. Consulte estos apartados para más información*:* Registrar FlowForce Server, Registrar MapForce Server, Registrar StyleVision Server y Registrar RaptorXML(+XBRL) Server.

Para más información lea el apartado Asignar licencias a productos registrados.

 © 2014 Altova GmbH

11.7.3 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.

11.7.4 Opciones de configuración

Temas de este apartado:

- Configuración de red
- Correos de alerta
- Mostrar información sobre cómo recibir e implementar licencias de evaluación

---

En la pestaña **Settings** (*imagen siguiente*) puede definir las opciones de configuración del servidor LicenseServer:

- la contraseña para iniciar sesión en LicenseServer. Escriba la contraseña y haga clic en el botón **Change Password** (*Cambiar contraseña*).
- opciones de red para la página de configuración web (*Web UI*), para el servidor proxy utilizado para conectarse a Internet si lo hay (*Proxy Server*) y para LicenseServer (*License Service*). Estas opciones se describen con más detalle debajo de la imagen.
- opciones del servidor de correo y el destinatario de correos de alerta a quien avisar cuando ocurra algo importante en LicenseServer. Estas opciones se describen con más detalle debajo de la imagen.
- puede probar estas opciones haciendo clic en el botón **Test Connection to Altova** (*Probar conexión con Altova*). Recuerde que debe guardar las opciones nuevas (con el botón **Save** situado al final del panel) antes de probar la conexión. El botón **Test Connection to Altova** está deshabilitado durante la prueba y se vuelve a habilitar cuando finaliza la prueba.

 © 2014 Altova GmbH

License Pool | Server Management | Server Monitoring | Settings | Messages(6) | Log Out | Help

LicenseServer Password

Change Password

Connectivity Test

Test connection to Altova.

Web UI

Configure the host addresses where the web UI is available to administrators.

- All interfaces and assigned IP addresses
- Local only (localhost)
- Only the following hostname or IP address: 0.0.0.0

Ensure this hostname or IP address exists or LicenseServer will fail to start!

Configure the port used for the web UI.

- Dynamically chosen by the operating system
- Fixed port 8088

Ensure this port is available or LicenseServer will fail to start!

Proxy Server

Configure the proxy server connection details if a proxy server is needed to communicate with Altova's servers.

Hostname: myproxy

Port Number: 1285

If the port number is left blank the default port 1080 will be used.

User Name: myusername

Password: •••••

Leave the user name and password blank if no authentication is required.

License Service

Configure the host addresses where the LicenseServer service is available to clients.

- All interfaces and assigned IP addresses
- Local only (localhost)
- Only the following hostnames or IP addresses:

Ensure the hostnames or IP addresses exist or LicenseServer will fail to start!

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**.

- *Proxy Server (disponible a partir de la versión 1.3):* si usa un servidor proxy para conectarse a Internet, los datos de este servidor proxy deben introducirse bajo este grupo de opciones (*imagen anterior*). Rellene estos campos si está usando un servidor proxy. Recuerde que muchos servidores proxy no exigen autenticación (con nombre de usuario y contraseña), en cuyo caso puede dejar en blanco estos dos campos. Para configurar LicenseServer para el servidor proxy, escriba el nombre de host del servidor proxy y un número de puerto, si es necesario.

- *License Service (servicio de licencias):* las direcciones IP pueden ser direcciones IP e interfaces del equipo o una dirección fija. Si incluye nombres de host y direcciones IP, sepárelas con comas y sin espacios (por ejemplo: `nombrehost1,direcciónIP1,nombrehost2`). El número de puerto está fijado en **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.

Realice una prueba de conectividad (*ver más arriba*) para comprobar si funcionan las opciones definidas.

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. En el grupo de opciones *Alert Mail* (*imagen anterior*) puede configurar los correos de alerta y la dirección de correo del administrador.

 © 2014 Altova GmbH

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* toma la dirección de la cuenta de correo electrónico desde la que se enviará el correo. 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. No olvide que estos eventos también se registran en la pestaña **Messages**.

Mostrar información sobre cómo recibir e implementar licencias de evaluación
Marque la casilla *Show hint how to receive evaluation licenses for a server product* situada al final de la pestaña **Settings** (*imagen anterior*) y al principio de la página de configuración aparecerán instrucciones para recibir e implementar licencias de evaluación.

11.7.5 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.

 © 2014 Altova GmbH

Índice

A

Altova LicenseServer,
ver LicenseServer, 526
Altova ServiceController, 533
API de Python,
API para XBRL, 284
API para XML, 205
API para XSD, 220
objeto Job, 204
Asignación de licencias en Linux, 26
Asignación de licencias en Mac OS X, 32
Asignación de licencias en Windows, 19
Asignar licencias, 557, 567

C

Cargar licencias, 544, 563
Catálogos, 34
Catálogos XML, 34
Cerrar sesión, 576
Comando de ayuda de la ILC, 124
Comandos de licencias en la ILC, 125
Comandos XQuery, 114
Comandos XSLT, 102
Comprobar el formato, 72
Configuración, 572
Configuración del servidor, 156
Contraseña,
predeterminada al iniciar LicenseServer, 537
Contraseña predeterminada, 537
Correos de alerta, 572

D

Documentos XQuery,
validación, 120
Documentos XSLT,
validación, 109

E

Ejecución de XQuery, 115
Ejemplo de script Python, 186, 191, 198
Extensiones de Altova,
funciones para gráficos, 462

F

FlowForce Server,
registrar con LicenseServer, 547
Funciones de extensión .NET,
campos de instancia, 517
campos estáticos, 516
constructores, 515
conversiones de tipos de datos, 518, 519
métodos de instancia, 517
métodos estáticos, 516
para XSLT y XQuery, 513
resumen, 513
tipos de datos .NET en XPath/XQuery, 519
tipos de datos XPath/XQuery en .NET, 518
Funciones de extensión .NET para XSLT y XQuery,
ver Funciones de extensión .NET, 513
Funciones de extensión en scripts MSXSL,
msxsl:script, 521
Funciones de extensión Java,
archivos de clases definidos por el usuario, 505
archivos JAR definidos por el usuario, 508
campos de instancia, 510
campos estáticos, 510
constructores, 509
conversiones de tipos de datos, 511, 512
métodos de instancia, 510
métodos estáticos, 510
para XSLT y XQuery, 504
resumen, 504
tipos de datos Java en XPath/XQuery, 512
tipos de datos XPath/XQuery en Java, 511
Funciones de extensión Java para XSLT y XQuery,
ver Funciones de extensión Java, 504
Funciones de extensión para XSLT y XQuery, 503
Funciones para gráficos,

© 2014 Altova GmbH

Funciones para gráficos,
ejemplo, 495
estructura de datos de gráficos, 490
fragmento de código, 486

I

Información de red, 528
Instalación, 14
Linux, 22
Mac OS X, 28
Windows, 15
Instalación en Linux, 23
Instalación en Mac OS X, 29
Instalación en Windows, 16
Interfaces,
resumen, 5
Interfaz .NET, 5
Interfaz COM, 5
Interfaz del administrador, 562
Interfaz HTTP, 5, 150
configuración del servidor, 156
preparar el servidor, 152
problemas de seguridad, 44
solicitudes cliente, 160
Interfaz Java, 5
Interfaz Python, 5, 180
crear scripts, 182
ejecutar scripts, 185

L

Licencias,
asignar, 557, 567
cargar, 544, 563
LicenseServer,
iniciar, 535
instalación en Linux, 530
instalación en Mac OS X, 532
instalación en Windows, 529
instrucciones para asignar licencias, 534
interfaz con, 562
opciones de configuración, 572
página de configuración, 562
registrar FlowForce Server con, 547
registrar MapForce Server con, 551
registrar StyleVision Server con, 553
Línea de comandos,
opciones, 130
resumen de uso, 46
y XQuery, 114
Linux,
asignación de licencias en, 26
instalación en, 23
Localización, 127

M

Mac OS X,
asignación de licencias en, 32
instalación en, 29
MapForce Server,
registrar con LicenseServer, 551
Mensajes, 576

O

Opciones de configuración de red, 572

P

Página de configuración, 562
abrir en Linux, 540
abrir en Mac OS X, 542
abrir en Windows, 537
URL de, 537
URL de (Linux), 540
URL de (Mac OS X), 542
Página de configuración de LicenseServer,
(ver Página de configuración), 540, 542
ver Página de configuración, 537
Página Gestión de servidores, 567
Página Supervisión de servidores, 571
Pestaña Gestión de servidores, 557
Problemas de seguridad, 44
Python,
problemas de seguridad, 44

© 2014 Altova GmbH

R

RaptorXML,
características, 8
ediciones e interfaces, 5
especificaciones compatibles, 10
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, 7
Recursos globales, 42
Registrar FlowForce Server con LicenseServer, 547
Registrar MapForce Server con LicenseServer, 551
Registrar StyleVision Server con LicenseServer, 553
Repertorio de licencias, 544, 563

S

Scripts en XSLT/XQuery,
ver Funciones de extensión, 503
ServiceController, 533
StyleVision Server,
registrar con LicenseServer, 553

T

Transformación XSLT, 103

V

Validación,
de cualquier documento, 68, 98
de documentos XQuery, 120
de documentos XSLT, 109
de DTD, 60
de instancias XBRL, 83
de instancias XBRL y taxonomías, 82
de instancias XML con DTD, 50
de instancias XML con XSD, 54
de taxonomías XBRL, 92
de XSD, 63
Validación XBRL,
ver Validación, 82

W

Windows,
asignación de licencias en, 19
instalación en, 16

X

XQuery,
funciones de extensión, 503
XSLT,
funciones de extensión, 503

© 2014 Altova GmbH