Altova MapForce 2024 Basic Edition

Manual del usuario y referencia

Altova MapForce 2024 Basic Edition
Manual del usuario y referencia

Fecha de publicación: 2024

Contenido

1 Introducción

Altova MapForce 2024 Basic Edition es una herramienta ETL potente para transformar e integrar datos. MapForce es una aplicación de 32/64 bits para Windows compatible con Windows 10, Windows 11 y Windows Server 2016 o superior. Las ediciones Enterprise y Professional están disponibles en 32 y 64 bits.

MapForce le permite convertir datos entre casi todos los formatos. MapForce tiene una interfaz gráfica 20 que incluye muchas opciones diferentes para gestionar, visualizar, manipular y ejecutar asignaciones por separado o como parte de proyectos de asignación. MapForce incluye una biblioteca amplia de funciones de procesamiento y conversión de datos 196 para filtrar y manipular datos de acuerdo con los requisitos de su proyecto de integración de datos.

En cuanto haya terminado de diseñar su asignación, puede obtener una vista previa del resultado en un panel independiente y guardar el resultado en la ubicación deseada. Aparte, puede generar código 66 para su ejecución externa.

Puede ampliar las funciones de MapForce integrando el programa con otros productos de Altova:

- Puede ejecutar sus asignaciones de datos usando MapForce Server. Esto le ayudará a automatizar las operaciones empresariales que requieren transformaciones de datos repetitivas. MapForce Server incluye un potente motor de transformación de datos y puede convertir todos los tipos de datos. Lo más importante es que se trata de un servidor multiplataforma que está disponible en Windows, macOS y Linux.
- RaptorXML Server es un motor ultrarápido que valida sus instancias.
- FlowForce Server le ayuda a automatizar sus tareas y le permite ejecutar sus asignaciones de datos como trabajos programados.
- StyleVision Server genera documentos de salida en los formatos HTML, RTF, PDF y Word.
- Puede utilizar StyleVision para diseñar hojas de estilos StyleVision Power (SPS) que permitan al servidor StyleVision Server generar resultados en varios formatos.
- DatabaseSpy es una herramienta versátil que permite diseñar, editar y consultar bases de datos.
- XMLSpy es particularmente útil si quiere editar sus archivos de asignación. Algunos cuadros de diálogo de MapForce le permiten abrir archivos directamente en XMLSpy.
- También puede usar MapForce como complemento de Microsoft Visual Studio y Eclipse. Así podrá acceder a las funciones de MapForce sin salir de su entorno de desarrollo favorito.

Última actualización: 09.04.2024

1.1 Novedades

En esta sección se describen las características y funciones de cada nueva versión de MapForce. Para más información consulte el apartado de la versión correspondiente.

1.1.1 Versión 2024

Versión 2024 Release 2

- Ahora es posible acceder a varios tutoriales en vídeo en el proyecto **`MapForceExamples`** (*ediciones Enterprise y Professional*). Además, puede añadir sus propios enlaces a recursos externos.
- Al implementar su asignación en FlowForce Server, puede elegir anexar los archivos de asignación para recuperarlos después. Esto evitará que pierda sus archivos de asignación y le permitirá descargarlos en cualquier momento (*ediciones Professional y Enterprise*).
- Los componentes de BD ahora pueden compartir la misma conexión de base de datos en tiempo de ejecución (*ediciones Enterprise y Professional*).
- Ahora es posible crear una asignación a partir de valores de tipos de enumeración en XML (*todas las ediciones*) y componentes XBRL (*Enterprise Edition*). Gracias a esta función le resultará más fácil asignar valores de enumeración: Ambos lados de la asignación de valores se rellenan previamente con todos los valores de enumeración y sólo tendrá que revisar y editar los valores relevantes de la asignación. Para más información consulte el apartado Asignación de valores 188.
- La generación de código C# es compatible con NET 8.0 (*ediciones Professional y Enterprise*). Para más información consulte el apartado Generación de código 66.
- Compatibilidad con mensajes FORTRAS EDI (*Enterprise Edition*).
- Compatibilidad con PostgreSQL 16, MySQL 8.2, MySQL 8.3, MariaDB 11.2, SQLite 3.45 (*ediciones Professional y Enterprise*).
- Mejoras y actualizaciones internas.

Versión 2024

- Otra utilidad nueva de MapForce que ya está disponible es PDF Extractor (*Enterprise Edition*). PDF Extractor sirve para crear plantillas de extracción PDF que puede importar a MapForce y utilizar como componentes de origen en las asignaciones de datos.
- En MapForce ahora es posible crear asignaciones de datos basadas en IA (*Enterprise Edition*). MapForce permite crear llamadas a servicios web (de tipo REST) a una API, como la API OpenAI, la API OpenAI de Azure, los servicios de IA de AWS, etc.
- Compatibilidad con SWIFT 2023 (*Enterprise Edition*).
- Ahora está disponible una nueva función de suspensión, **`sleep`**, que permite pasar datos tras un retraso especificado (*Ediciones Enterprise y Professional*).
- Las bases de datos MySQL y MariaDB ahora tienen compatibilidad nativa (*ediciones Professional y Enterprise*).
- Se ha mejorado y ampliado la funcionalidad de las conexiones de secundarios equivalentes, incorporando nuevas opciones de combinación. Para más detalles, consulte Conectar los secundarios equivalentes 53.
- Además del tipo de concesión *Código de autorización*, los tipos *Credenciales de cliente* y *Credenciales de contraseña del propietario de recursos* ahora también son compatibles con las credenciales OAuth (*Enterprise Edition*).

- Mejoras y actualizaciones internas.

1.1.2 Versión 2023

Versión 2023 Release 2

- Ahora puede generar la declaración `standalone="yes"` en la declaración XML de los archivos XML de destino. Para más detalles consulte Configuración de componentes XML [117].
- Hemos reorganizado la Ayuda [481] para que pueda acceder a la Ayuda en línea de forma predeterminada, aunque sigue teniendo la opción de usar el archivo PDF local [472].
- Ahora puede añadir comentarios de estilo post-it a una asignación. Para más información consulte Opciones [34].
- Hemos añadido opciones de configuración de red [476].
- Ahora MapForce también es compatible con mensajes VDA EDI (*Enterprise Edition*).
- Mejoras y actualizaciones internas.

Versión 2023

- Hemos añadido compatibilidad con estos temas: *Clásico*, *Ligero* y *Oscuro*. Visite el apartado Ventanas [480] para obtener más información.
- Mejoras y actualizaciones internas.
- Hemos ampliado la compatibilidad con Eclipse a las versiones: 2022-09, 2022-06, 2022-03, 2021-12 (*ediciones Professional y Enterprise*).
- Compatibilidad con mensajes ODETTE EDI (*Enterprise Edition*).
- Compatibilidad con la Especificación XII Transformation Registry 5 (*Enterprise Edition*).
- Ahora es posible crear parámetros en funciones definidas por el usuario [213] basados en BD y variables [161] con una estructura de tablas relacionadas (*ediciones Professional y Enterprise*).
- Ahora es posible enviar estructuras de solicitud `application/x-www-form-urlencoded` a un servicio REST (*Enterprise Edition*).
- Compatibilidad con los directorios UN/EDIFACT D.21B y D.22A (*Enterprise Edition*).
- Compatibilidad con SQLite 3.39.2, MariaDB MySQL 10.9.2, PostgreSQL 14.5 (*ediciones Professional y Enterprise*).
- Compatibilidad con el Gestor de esquemas XML [130], con el que podrá instalar y administrar esquemas XML de forma centralizada para todas las aplicaciones de Altova compatibles con XBRL.
- Compatibilidad con delimitadores EDI asignables (*Enterprise Edition*). Esta funcionalidad es compatible actualmente con estos estándares EDI: EDIFACT, X12 y NCPDP SCRIPT.

1.1.3 Versión 2022

Versión 2022 Release 2

- Actualizaciones y mejoras internas
- Hemos ampliado la compatibilidad con Eclipse a las versiones: 2021-12, 2021-09; 2021-06; 2021-03 (*ediciones Professional y Enterprise*).
- El complemento de MapForce para Visual Studio y la generación de código son compatibles con Visual Studio 2022 (*ediciones Professional y Enterprise*).

- La generación de código es compatible con NET 6.0 (*ediciones Professional y Enterprise*).
- También hemos ampliado la compatibilidad con bases de datos a: PostgreSQL 14, SQLite 3.37.2, MariaDB 10.6.5, MySQL 8.0.28, IBM DB2 11.5.7 (*ediciones Professional y Enterprise*).
- Ahora se puede generar una vista previa de las imágenes en la ventana **Proyecto** (*ediciones Professional y Enterprise*).
- Ahora es posible crear indicadores de tramitación que son conformes con la EBA para los componentes XBRL de destino (*Enterprise Edition*).

Versión 2022

- Actualizaciones y mejoras internas
- Hemos ampliado la compatibilidad con Eclipse a las versiones: 2021-09; 2021-06; 2021-03; 2020-12 (*ediciones Professional y Enterprise*).
- Las conexiones de copia total[55] ahora también son compatibles con JSON. Esta funcionalidad sólo está disponible para tipos compatibles con JSON (*Enterprise Edition*).
- En StyleVision existe un panel de resultados nuevo llamado *Texto*. Si adjunta un archivo SPS a un componente puede acceder a una vista previa del formato de salida en texto plano en MapForce (*ediciones Professional y Enterprise*).
- Compatibilidad con JSON Schema en variables[160] y parámetros[211] (*Enterprise Edition*).
- Compatibilidad con bases de datos NoSQL: MongoDB y CouchDB (*Enterprise Edition*).
- Ahora hay disponible una biblioteca de funciones `bson` que permite crear y manipular tipos BSON (*Enterprise Edition*).
- Compatibilidad con los directorios EDIFACT/ONU D.20B y D.21A.
- Compatibilidad con SWIFT 2021.

1.1.4 Versión 2021

Versión 2021 Release 3

- Compatibilidad ampliada para JSON Schema Draft 2019-09 y Draft 2020-12. (*sólo en la edición Enterprise Edition*).

Versión 2021 Release 2

- Ahora se puede usar XSLT 3.0 como lenguaje de asignación, véase Generar código XSLT[66]. Además, MapForce ahora incluye funciones integradas nuevas compatibles con XSLT 3.0, véase la referencia de la biblioteca de funciones[235].
- Otras mejoras y actualizaciones internas.

Versión 2021

- Otras mejoras y actualizaciones internas.

1.1.5 Versión 2020

Versión 2020 Release 2

- La ventana nueva Gestionar bibliotecas [200] permite ver y gestionar todas las bibliotecas de funciones importadas por la asignación actual, así como gestionarlas a nivel de programa (esto incluye las funciones de MapForce definidas por el usuario y otros tipos de biblioteca). Por ejemplo, puede copiar y pegar fácilmente funciones definidas por el usuario de una asignación a otra (véase Copiar y pegar funciones definidas por el usuario entre asignaciones [210]).
- Cuando un archivo de asignación importa bibliotecas, la ruta de los archivos de biblioteca importados es relativa a la asignación por defecto (véase Rutas relativas de acceso a bibliotecas [203]).
- Si un archivo de asignación importa bibliotecas XSLT, se puede generar código XSLT que use una ruta relativa para hacer referencia a los archivos de biblioteca importados. Puede encontrar esta opción en el cuadro de diálogo "Configurar asignación" [76].
- Otras mejoras y actualizaciones internas

Versión 2020

- Cuando reemplace valores con ayuda de una tabla de consulta puede pegar datos tabulares (pares clave-valor) desde un origen externo, como CSV o Excel, en la asignación. Además, ahora es más fácil manejar los casos en los que no se encuentra un valor en la tabla de consulta predefinida (para procesar esos valores ya no es necesario usar la función **`subsitute-missing`**. Consulte Usar asignaciones de valores [184].
- Otras mejoras y actualizaciones internas.

1.2 ¿Qué es MapForce?

Sitio web de Altova: Herramienta de asignación de datos

MapForce es una potente herramienta gráfica que permite convertir e integrar cualquier combinación de datos. Consulte Asignación: origen y destino [16] para ver una lista completa de los formatos disponibles. Una asignación consiste en uno o más orígenes de datos y uno o más destinos de datos [32]. La asignación puede incluir uno o más componentes de transformación [33] con los que cuenta un amplio número de opciones de procesamiento y filtrado. Para saber más sobre diferentes tipos de asignaciones, consulte Tipos de asignaciones [17] y Tutoriales [78].

Para poder ejecutar una asignación debe indicar una estructura de datos para los archivos de origen y de destino. Por ejemplo, un esquema XML define la estructura de un archivo XML. Los datos se asignan (del origen al destino) mediante acciones de arrastrar y colocar en la interfaz gráfica del usuario. No es necesario que escriba código de programa para la asignación, ya que MapForce lo genera automáticamente. Puede usar este código para transformar documentos con la estructura de datos de origen en documentos con la estructura de datos de destino.

Todas las ediciones de MapForce tienen una versión de 32 bits. Además, las ediciones Enterprise y Professional están disponibles en 64 bits.

Modelo abstracto

El modelo abstracto que ve a continuación ilustra una de las transformaciones de datos básicas que puede realizar con MapForce. El esquema de origen describe la estructura de la instancia de origen. El esquema de destino describe la estructura de la instancia de destino. En función de sus necesidades los esquemas de origen y de destino pueden tener la misma estructura o estructuras distintas. Al conectar el origen y el destino, la asignación genera código de transformación (en el lenguaje de transformación [17] seleccionado) que lee datos de la instancia de origen y los escribe en la de destino. Para ver cómo se implementa este modelo de transformación de datos en un ejemplo concreto, consulte el Tutorial 1 [79].

En situaciones reales, puede mezclar y unir cualquier combinación de fuentes de datos (p.ej., XML, EDI y archivos de texto) y asignarlos a cualquier combinación de destinos de datos (p.ej., una base de datos y un archivo Excel).

Notas generales

Puede encontrar los archivos de asignación que se usan y a los que se hace referencia en el manual en estas ubicaciones:

- `C:\Usuario\<usuario>\Documentos\Altova\MapForce2024\MapForceExamples`
- `C:\Usuario\<usuario>\Documentos\Altova\MapForce2024\MapForceExamples\Tutorial`
- `C:\Usuario\<usuario>\Documentos\Altova\MapForce2024\MapForceExamples\Tutorial\BasicTutorials`

En esta sección

En esta sección encontrará:

1.2.1 Asignación: origen y destino

En MapForce, origen y destino son términos esenciales que hacen referencia a las estructuras de datos desde o hacia las que se asignan los datos. Estas son las tecnologías que puede usar como origen o destino de las asignaciones de datos en MapForce:

MapForce Basic Edition

- XML y XML Schema

MapForce Professional Edition

- XML y XML Schema
- Archivos planos, incluido el formato CSV (valores separados por comas) y FLF (campo de longitud fija)
- Bases de datos (las principales bases de datos relacionales)
- Archivos binarios (contenido BLOB sin formato)

MapForce Enterprise Edition

- XML y XML Schema
- Archivos planos, incluido el formato CSV (valores separados por comas) y FLF (campo de longitud fija)
- Datos de archivos de texto heredados pueden asignarse y convertirse a otros formatos con la aplicación MapForce FlexText
- Bases de datos SQL: las principales bases de datos relacionales
- Bases de datos NoSQL
- Archivos binarios (contenido BLOB sin formato)
- Estándares EDI
- Archivos JSON
- Archivos de Microsoft Excel 2007 y posteriores
- Archivos de instancia y taxonomías XBRL
- Protocol Buffers

- Archivos PDF basados en plantillas PDF creadas en PDF Extractor (sólo se pueden usar como fuentes de datos)

1.2.2 Tipos de asignaciones

Sitio web de Altova: Vídeos de demostración de MapForce

El grado de complejidad de una asignación puede variar en función de sus necesidades o requisitos empresariales: Por ejemplo, puede que necesite configurar su asignación para (i) leer datos de un solo origen y escribir estos datos en varios destinos o para (ii) combinar datos de varios orígenes en un solo destino. Se pueden usar distintas estructuras de datos como origen y destino: p.ej. archivos XML, bases de datos, archivos EDI, etc. Para obtener más información sobre formatos de origen y destino consulte Asignación: origen y destino 16

La complejidad de los diseños de asignación incluye pero no se limita a:

- Asignar un archivo de origen a un archivo de destino. Visite Tutorial 1 79 para obtener más información.
- Combinar varios orígenes de datos en un único destino. Visite Tutorial 2 89 para obtener más información.
- Asignar datos de un solo origen al primer archivo de destino y, a continuación, filtrar los datos de forma que sólo un subconjunto de estos se asigne al segundo archivo de destino. Consulte Tutorial 3 94.
- Asignar varios archivos de origen a varios archivos de destino. Consulte Tutorial 4 103.

Independientemente de la tecnología que utilice, MapForce determina automáticamente la estructura de los datos o sugiere que se asigne un esquema a esos datos. MapForce también puede generar esquemas a partir de un archivo de instancia de muestra. Por ejemplo, si está trabajando con un archivo de instancia XML pero no tiene una definición de esquema, MapForce puede generar uno. De esta forma MapForce hace que los datos del archivo XML se puedan asignar a otros archivos o formatos. Para más información sobre los términos y las funcionalidades básicas de MapForce, consulte Fundamentos de la asignación de datos 30 e Interfaz del usuario 20.

Proyectos (ediciones Professional y Enterprise)
Para administrar y acceder a los diseños de las asignaciones de datos de forma más fácil puede organizarlos en proyectos. Además de generar código para cada asignación individual en el proyecto, podrá generar código de programa para el proyecto entero.

1.2.3 Lenguajes de transformación

En MapForce los lenguajes de transformación sirven para generar el código de transformación que ejecuta las asignaciones. Puede seleccionar o modificar el lenguaje de transformación en cualquier momento. También puede generar código de programa con el comando de menú **Archivo | Generar código en** o **Archivo | Generar código en el lenguaje seleccionado** y usar ese código para llevar a cabo transformaciones de datos fuera de MapForce. Para más información consulte Generación de código 66.

Según la edición de MapForce puede elegir uno de estos lenguajes para la transformación de datos:

Basic Edition	ediciones Enterprise y Professional
• XSLT 1.0 • XSLT 2.0 • XSLT 3.0	• XSLT 1.0 • XSLT 2.0 • XSLT 3.0 • BUILT-IN • XQuery • Java • C# • C++

Si selecciona XSLT 1-3 o XQuery como lenguaje de transformación, podrá ver el código de transformación en un panel separado de MapForce.

Si quiere seleccionar todas las tablas, tiene dos opciones:

- En el menú **Resultados** haga clic en el nombre del lenguaje que quiere usar para la transformación.
- Haga clic en el lenguaje en la barra de herramientas de **selección de lenguajes** (*véase más abajo*).

Si selecciona otro lenguaje de transformación, ciertas características de MapForce pueden ser incompatibles. Para saber más consulte Notas sobre compatibilidad 487.

Durante el proceso de diseño o la vista previa, MapForce valida constantemente la integridad de los esquemas y de las transformaciones. Si hay errores de validación, los muestra en la ventana Mensajes 25. Esto puede resultar muy útil ya que permite revisar y corregir esos errores al instante.

BUILT-IN
Cuando se selecciona el lenguaje de transformación integrado BUILT-IN, MapForce utiliza su motor de transformación nativo para ejecutar las asignaciones. MapForce también usa implícitamente esta opción cada vez que se genera la vista previa de los resultados de asignaciones que usan Java, C# o C++ como lenguaje de transformación.

El motor integrado BUILT-IN ejecuta asignaciones sin necesidad de procesadores externos, por lo que puede ser una buena elección si no dispone de mucha memoria. Si no necesita generar código de programa en un lenguaje concreto, use el lenguaje de transformación integrado como opción predeterminada, ya que es compatible con la mayoría de funciones de MapForce, al contrario que muchos otros lenguajes, como se explica en Notas sobre compatibilidad 487. Además, si selecciona el motor integrado como lenguaje de transformación, puede automatizar sus asignaciones en MapForce Server. Para obtener más información consulte Automatizar asignaciones con productos de Altova 427.

1.2.4 Integración con productos de Altova

Puede ejecutar transformaciones en MapForce con los motores XSLT/XQuery integrados. MapForce también se puede usar en combinación con otros productos de Altova (*véase más abajo*).

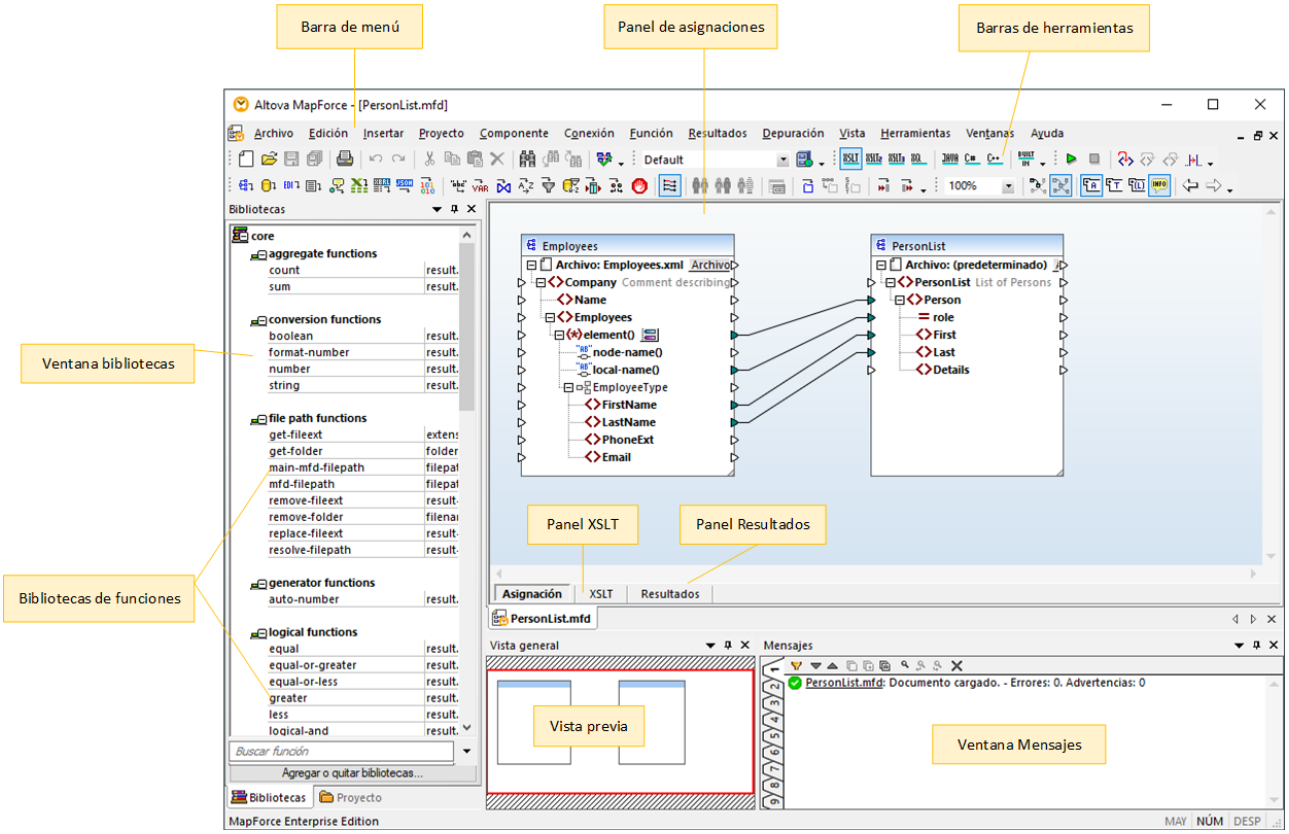
XMLSpy
Si tiene XMLSpy instalado en el mismo equipo que MapForce, podrá abrir y editar cualquier tipo de archivo compatible en XMLSpy desde MapForce directamente Por ejemplo, si hace clic con el botón derecho en un componente XML de MapForce, el menú contextual incluye el comando **Componente | Editar la definición del esquema en XMLSpy**.
RaptorXML Server
Puede ejecutar el código XSLT generado directamente en MapForce y obtener una vista previa del resultado de la transformación de datos. Cuando necesite un mayor rendimiento, puede procesar la asignación con RaptorXML Server, un rapidísimo motor de transformación XML.
MapForce Server (ediciones Enterprise y Professional)
Puede automatizar tareas de MapForce con Altova MapForce Server, que se puede instalar en sistemas Windows, Linux y macOS. MapForce Server permite ejecutar las transformaciones especificadas en una asignación mediante llamadas a la API (.NET, COM, Java) además desde la línea de comandos del sistema operativo correspondiente.
FlowForce Server (ediciones Enterprise y Professional)
También automatizar tareas de MapForce con Altova FlowForce Server, que se puede instalar en sistemas Windows, Linux y macOS. FlowForce Server permite ejecutar tareas de MapForce Server de forma planificada.
StyleVision (ediciones Enterprise y Professional)
Con StyleVision podrá usar hojas de estilos de StyleVision o diseñar hojas nuevas para obtener vistas previas del resultado de la transformación de datos en formato HTML, RTF, PDF o Word 2007+.
MapForce como complemento
La edición Enterprise y Professional de MapForce se puede instalar como complemento de Visual Studio y Eclipse. Así podrá diseñar asignaciones y acceder a las funciones de MapForce sin salir de su entorno de desarrollo favorito.

Para obtener más información sobre la automatización de tareas consulte Automatizar asignaciones con productos de Altova 427.

1.3 Interfaz del usuario

La interfaz gráfica del usuario de MapForce está organizada como un entorno de desarrollo integrado. A continuación puede ver los componentes principales de la interfaz. Además, puede cambiar la configuración de la interfaz con el comando de menú **Herramientas | Personalizar**. Todas las ventanas de la interfaz incluyen los botones , que sirven para mostrar, ocultar, anclar o acoplar las ventanas. Puede restaurar todas las barras de herramientas a su estado predeterminado con sólo ejecutar el comando de menú **Herramientas | Restaurar barras de herramientas y ventanas**.

La imagen siguiente muestra las partes principales de la interfaz gráfica del usuario de MapForce.



Para más información las características y funciones de cada parte consulte el apartado correspondiente.

Apartados de esta sección

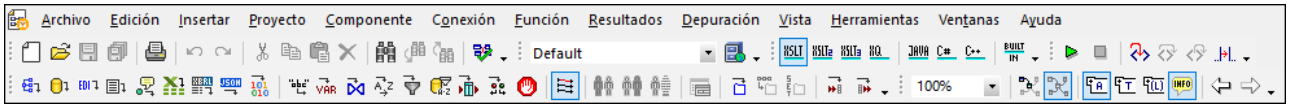
Esta sección tiene varias partes:

1.3.1 Barras de herramientas

En este apartado explicamos las distintas barras de herramientas.

Barra de menú y barras de herramientas

La **barra de menú** muestra los diferentes menús de la aplicación. Por su parte, las barras de herramientas muestran un grupo de botones que corresponden a comandos de MapForce. Las barras de herramientas se pueden arrastrar a una posición diferente si lo desea. En la imagen siguiente puede ver la **barra de menú** y las barras de herramientas. La interfaz actual depende de la edición de MapForce que esté usando y de cómo configure la aplicación.



Barra de estado de la aplicación

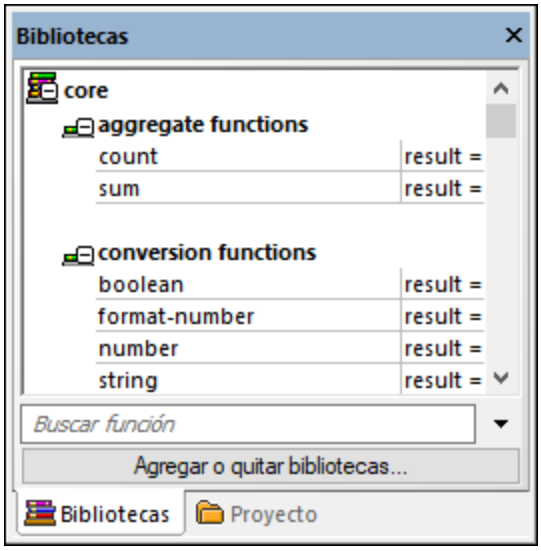
La barra de estado está situada en la parte inferior de la ventana de la interfaz y muestra información general sobre MapForce. Si pasa el puntero del ratón sobre los botones verá aparecer información rápida sobre ellos. Si usa la versión de 64 bits de MapForce, el nombre de la aplicación que aparece en la barra de estado incluye el sufijo x64. La versión de 32 bits no lleva ningún sufijo.

1.3.2 Ventanas

En este apartado explicamos las distintas ventanas.

Ventana Bibliotecas

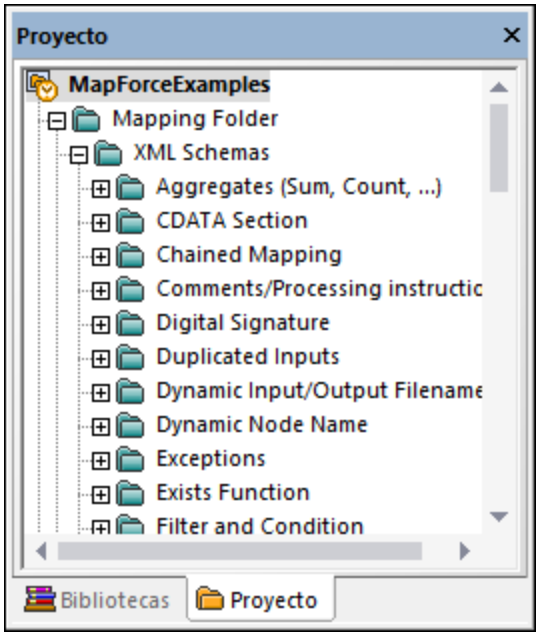
En la ventana **Bibliotecas** aparecen todas las funciones integradas de MapForce, organizadas por biblioteca. La lista de funciones que aparecen en esta ventana depende del lenguaje de transformación elegido en el menú **Resultados** o en la barra de herramientas de **selección de lenguajes**. Para saber más consulte Seleccionar el lenguaje de transformación 17. Además, en esta ventana aparecen también las funciones definidas por el usuario que se hayan creado y las bibliotecas externas que se hayan importado.



Para buscar funciones por nombre o por su descripción, escriba el valor de búsqueda en el cuadro de texto situado en la parte inferior de la ventana **Bibliotecas**. Para buscar todas las instancias de una función dentro de la asignación activa basta con hacer clic con el botón derecho en la función y elegir el comando **Buscar todas las llamadas** en el menú contextual. En la ventana **Bibliotecas** también puede ver el tipo de datos de la función y su descripción. Véase Funciones 196 para obtener más información.

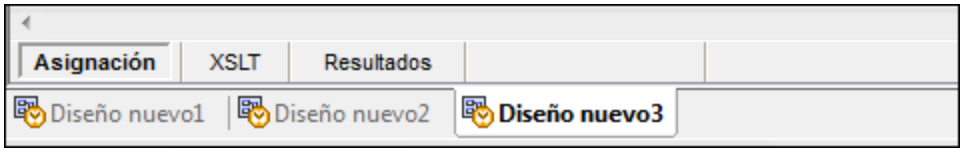
Ventana de proyecto

MapForce ofrece una interfaz compatible con múltiples documentos y permite agrupar las asignaciones en proyectos. La ventana **Proyecto** muestra todos los archivos y todas las carpetas del proyecto. La extensión de los archivos de proyecto de MapForce es **`*.mfp`**. Cuando quiera buscar una asignación dentro de un proyecto, haga clic dentro de la ventana **Proyecto** y pulse **Ctrl+F**.



Ventana(s) de asignación

MapForce usa una interfaz de documentos múltiples. Cada uno de los archivos de asignación que abre en MapForce cuenta con una ventana separada. Esto le permite trabajar con varias ventanas de asignación y colocarlas y cambiar sus tamaños como necesite, todo dentro de la ventana principal de MapForce. También puede colocar todas las ventanas abiertas con las opciones estándar de Windows: En mosaico horizontal, en mosaico vertical y en cascada. Cuando hay varias asignaciones abiertas en MapForce puede cambiar rápidamente de una a otra con las pestañas que hay bajo el panel Asignación (imagen siguiente).



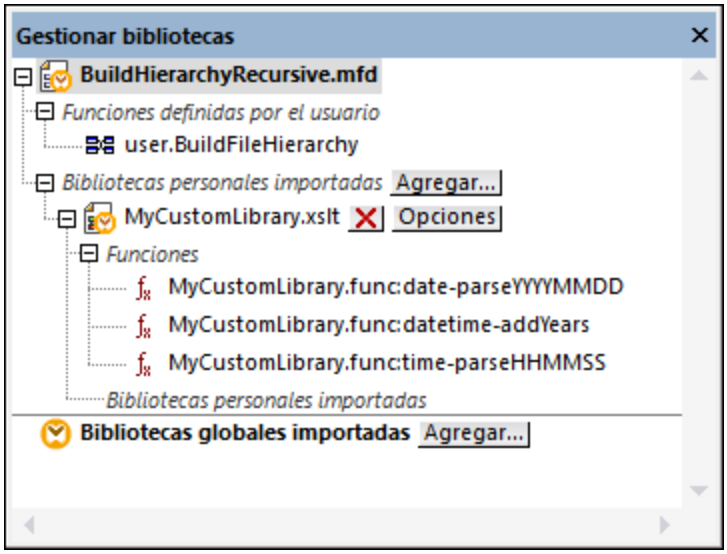
Puede acceder a las opciones de configuración de la ventana con el comando de menú **Ventanas | Ventanas**. El cuadro de diálogo "Ventanas" permite llevar a cabo distintas acciones, como activar, guardar, cerrar o minimizar las ventanas de asignación abiertas. Para seleccionar varias ventanas en el cuadro de diálogo "Ventanas" haga clic en las entradas correspondientes mientras mantiene pulsada la tecla **Ctrl**.

Administrar la ventana Bibliotecas

En esta ventana puede ver y gestionar todas las funciones definidas por el usuario (UDFs), así como las bibliotecas personalizadas que haya importado que estén usando las asignaciones abiertas en ese momento.

Por defecto, la ventana Gestionar bibliotecas no está visible. Para que aparezca tiene dos opciones:

- En el menú **Vista** haga clic en **Gestionar bibliotecas**.
- En la parte inferior de la ventana Bibliotecas haga clic en **Agregar o quitar bibliotecas**.



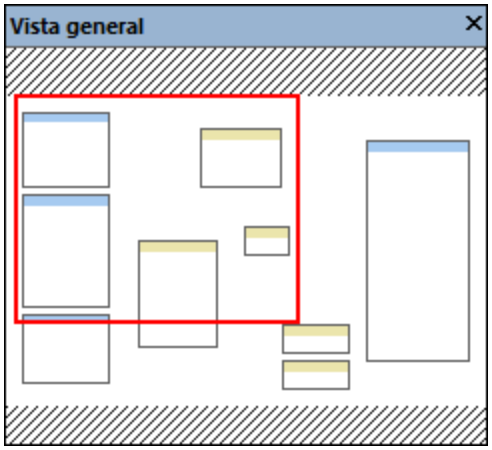
Puede elegir si quiere ver solamente las funciones definidas por el usuario del archivo de asignación activo o si quiere ver las de todos los archivos de asignación abiertos. Para ver las funciones y bibliotecas importadas para todas las asignaciones abiertas actualmente haga clic dentro de la ventana y seleccione **Mostrar documentos abiertos** en el menú contextual.

Si en vez de el nombre del archivo de asignación abierto prefiere ver su ruta de acceso haga clic dentro de la ventana y seleccione **Mostrar todas las rutas de acceso** en el menú contextual.

Para más información consulte Gestionar bibliotecas de funciones 200.

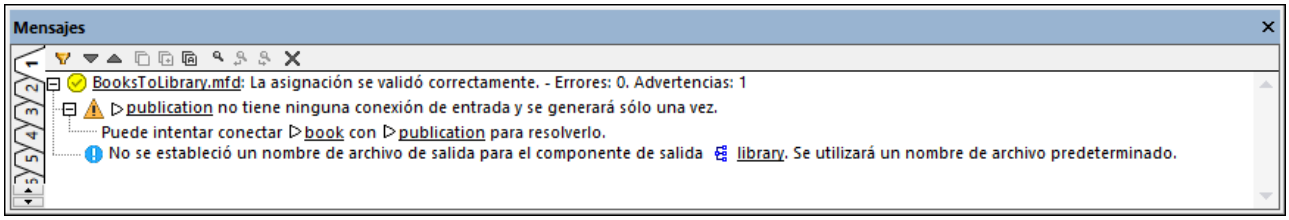
Ventana Vista general

La ventana **Vista general** ofrece una perspectiva general del panel Asignación 26. Desde esta ventana podrá navegar más fácilmente por el área de asignación cuando la asignación sea muy grande. Para navegar por la asignación basta con arrastrar el rectángulo rojo por la ventana.



1.3.3 Ventana Mensajes

La ventana **Mensajes** (*imagen siguiente*) muestra el estado de validación, mensajes, errores y/o advertencias cuando se accede a la vista previa de una asignación o cuando esta se valida 64. Para ir hasta el componente o estructura que provocó el mensaje, el error o la advertencia haga clic en el texto subrayado en la ventana **Mensajes**.



Iconos de estado de validación

Para validar la asignación MapForce comprueba, entre otras muchas cosas, si faltan conexiones o si hay conexiones incorrectas, si hay tipos de componente incompatibles, etc. El resultado de la validación aparece en la **ventana Mensajes** acompañado de uno de estos iconos de estado:

Icono	Significado
	La validación finalizó correctamente.
	La validación finalizó con advertencias.
	Error de validación.

La ventana **Mensajes** también puede mostrar mensajes de información, advertencia y error:

Icono	Significado
	Mensaje de información. La ejecución de la asignación no se interrumpe.
	Mensaje de advertencia. La ejecución de la asignación no se interrumpe. Pueden aparecer cuando, por ejemplo, no se crean conexiones obligatorias con conectores de entrada. Cuando esto ocurra, se generarán resultados para los componentes que tengan conexiones válidas.
	Mensaje de error. La ejecución de la asignación se interrumpe y no se generan resultados. Tampoco se puede generar una vista previa del código XSLT o XQuery.

Para ir hasta el componente o estructura que provocó el mensaje, el error o la advertencia haga clic en el texto subrayado en la ventana **Mensajes**.

Acciones relacionadas con los mensajes

Estas son las acciones que puede completar desde la ventana **Mensajes**:

Icono	Descripción
	Filtrar los mensajes según el nivel de gravedad: información, errores y advertencias. Seleccione **Activar todos** para incluir todos los niveles de gravedad (opción predeterminada). Seleccione **Desactivar todos** para quitar todos los niveles de gravedad del filtro. Cuando seleccione esta opción, solamente se mostrarán mensajes relacionados con la ejecución o con el estado de validación.
	Ir a la línea siguiente.
	Ir a la línea anterior.
	Copiar la línea seleccionada en el portapapeles.
	Copiar la línea seleccionada en el portapapeles, incluidas todas sus líneas anidadas.
	Copiar todo el contenido de la ventana Mensajes en el portapapeles.
	Buscar un texto en la ventana Mensajes. Ofrece una opción para buscar **Solo palabras completas** y la opción **Coinc. mayús/min** para refinar la búsqueda.
	Buscar un texto desde la línea seleccionada hasta el final de los mensajes.
	Buscar un texto desde la línea seleccionada hasta el principio de los mensajes.
	Borrar el contenido de la ventana Mensajes.

Cuando trabaje con varios archivos de asignación a la vez, puede consultar los mensajes de información, advertencia y error de cada archivo en pestañas diferentes. Para ello, antes de validar la asignación, seleccione la pestaña numerada correspondiente en el lateral izquierdo de la ventana **Mensajes**.

1.3.4 Paneles

En este apartado explicamos los distintos paneles.

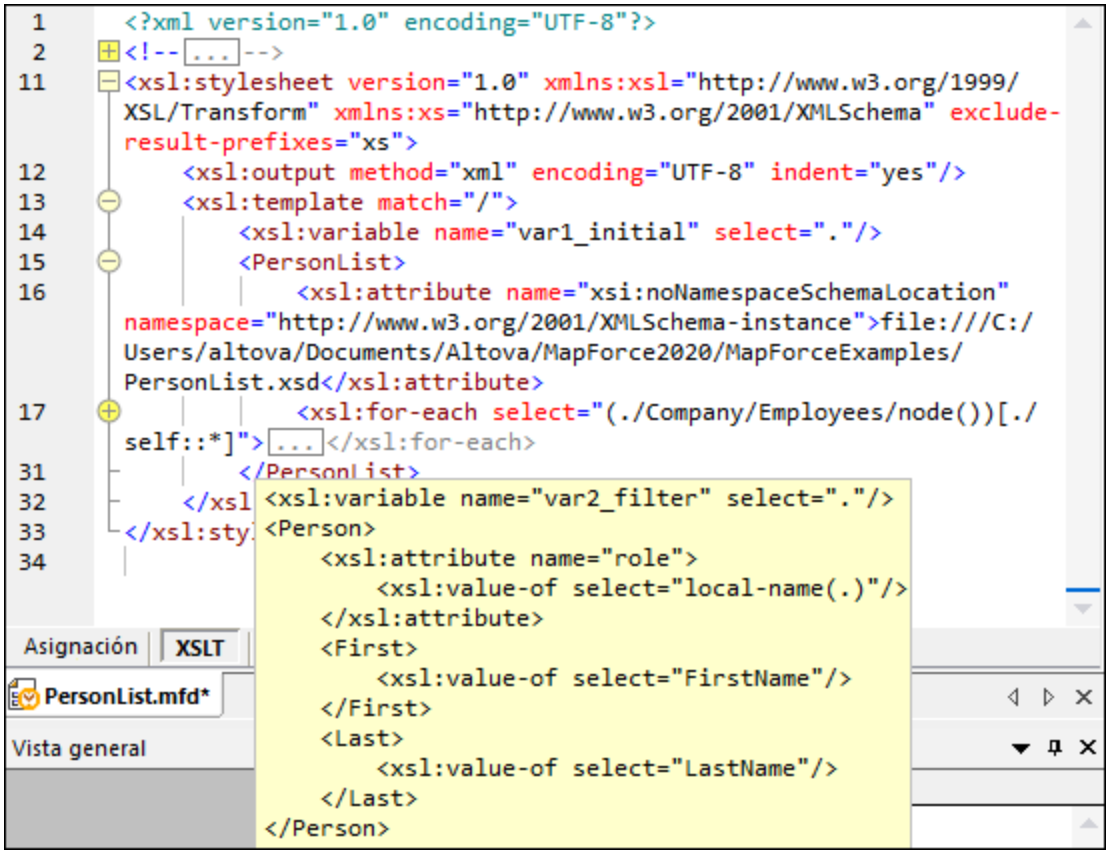
Panel Asignación

El panel **Asignación** es el área de trabajo donde se diseñan las asignaciones[64]. En este panel puede añadir componentes de asignación (archivos, esquemas, constantes, variables, etc.) desde el menú **Insertar**. Para más información consulte Agregar componentes a la asignación[36]. También puede arrastrar funciones desde la ventana **Bibliotecas** hasta el panel **Asignación**. Para más información consulte Funciones[197].

Panel XSLT

El panel **XSLT** muestra el código de transformación XSLT 1.0 que se genera a partir de la asignación. Para cambiar a este panel seleccione el lenguaje de transformación[17] XSLT (XSLT, XSLT 2 o XSLT3) y después haga clic en la pestaña del mismo nombre.

Este panel incluye funciones de numeración de líneas y plegamiento de código. Para expandir y contraer porciones de código basta con hacer clic en lo iconos + y - situados en el lateral izquierdo del panel. Las porciones de código contraído se señalan con puntos suspensivos. Para ver el código contraído debe pasar el cursor del ratón sobre los puntos suspensivos. Al hacerlo se abre un cuadro emergente que muestra una vista previa del código, como se ve en la imagen siguiente. No olvide que si la vista previa no cabe en el cuadro emergente, al final del código aparecen otra vez puntos suspensivos.



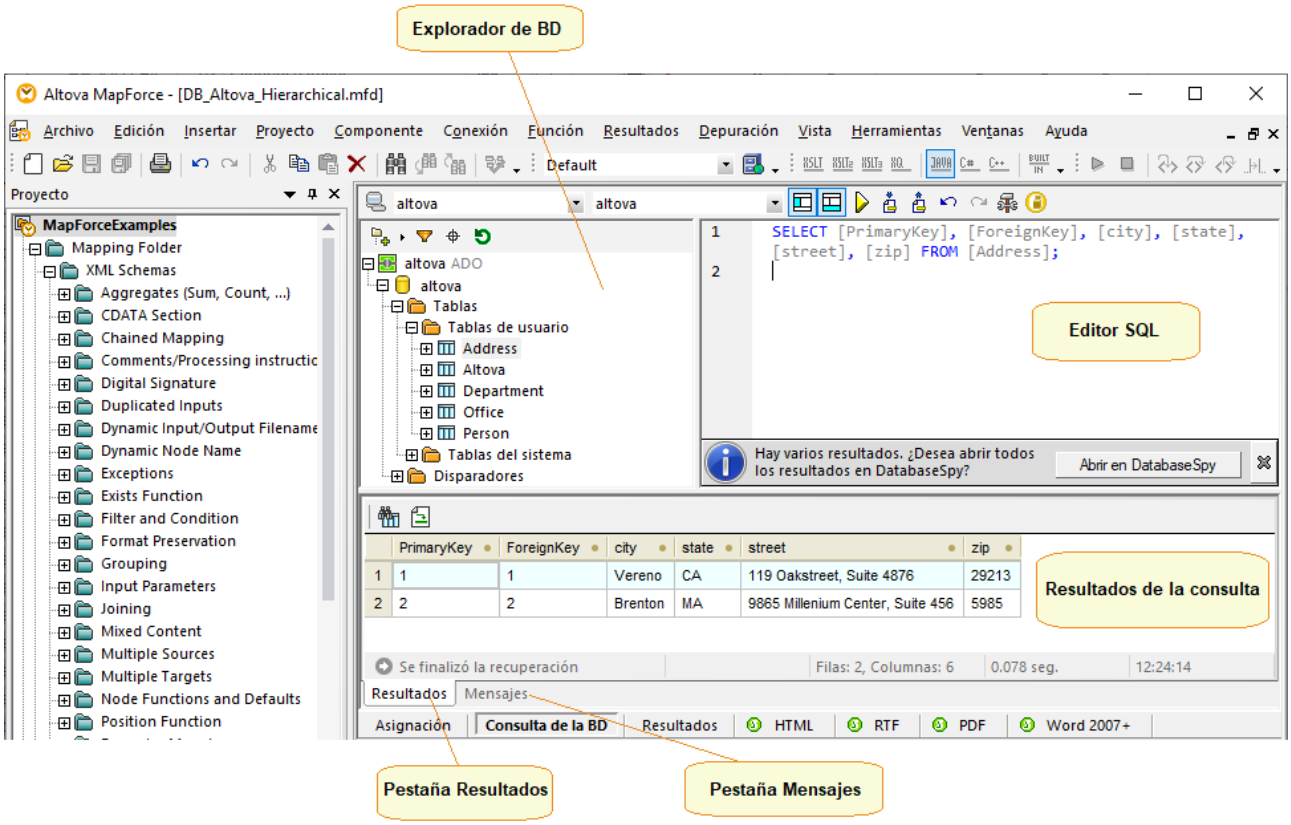
Para configurar las opciones de la vista, como la sangría, los marcadores de final de línea, etc. Haga clic con el botón derecho en el panel y seleccione **Configurar la vista Texto** en el menú contextual. También puede hacer clic en (**Configurar la vista Texto**) en la barra de herramientas.

Panel XQuery (ediciones Enterprise y Professional)

En el panel **XQuery** puede ver el código de transformación generado a partir de la aplicación al hacer clic en el botón **XQuery** . Este panel se habilita si selecciona XQuery como lenguaje de transformación. Este panel también incluye funciones de numeración de líneas y plegamiento de código idénticas a las del panel XSLT (véase más arriba).

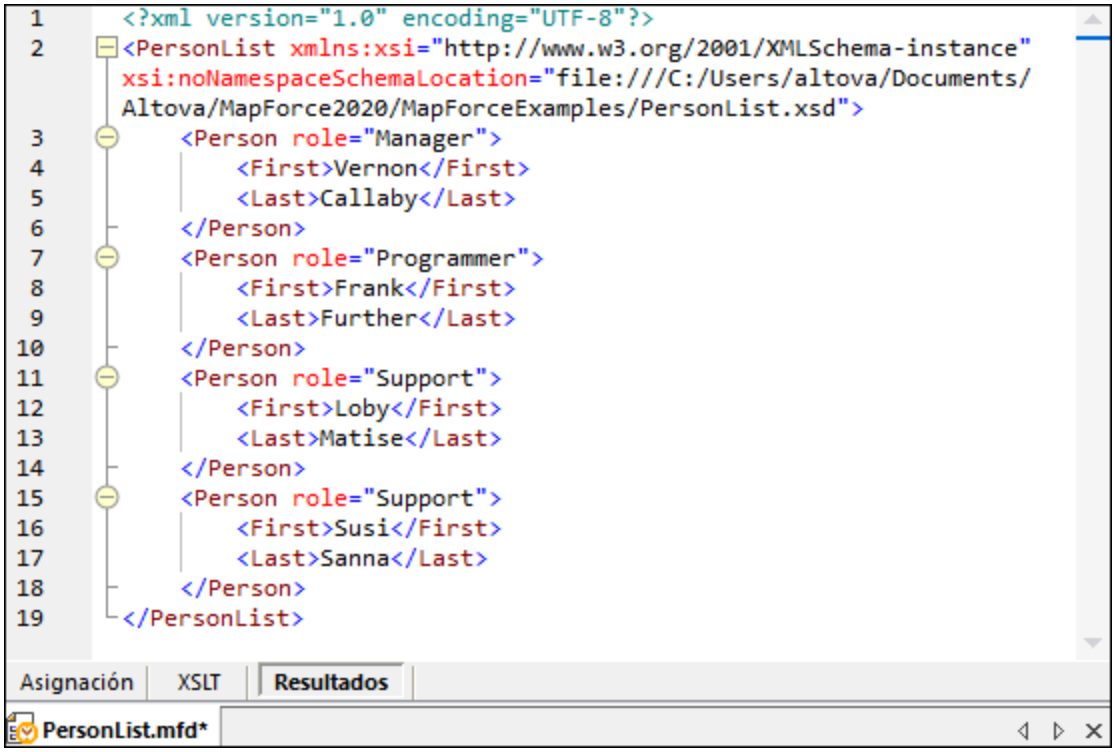
Panel Consulta de BD (ediciones Enterprise y Professional)

El **panel de consulta de DB** sirve para consultar las principales bases de datos. En este panel puede trabajar con varias conexiones activas a diferentes bases de datos.



Panel Resultados

El panel **Resultados** muestra el resultado de la transformación de datos. Si la asignación genera varios archivos podrá navegar por ellos de forma secuencial.



Este panel también incluye funciones de numeración de líneas y plegamiento de código idénticas a las del panel XSLT (véase más arriba).

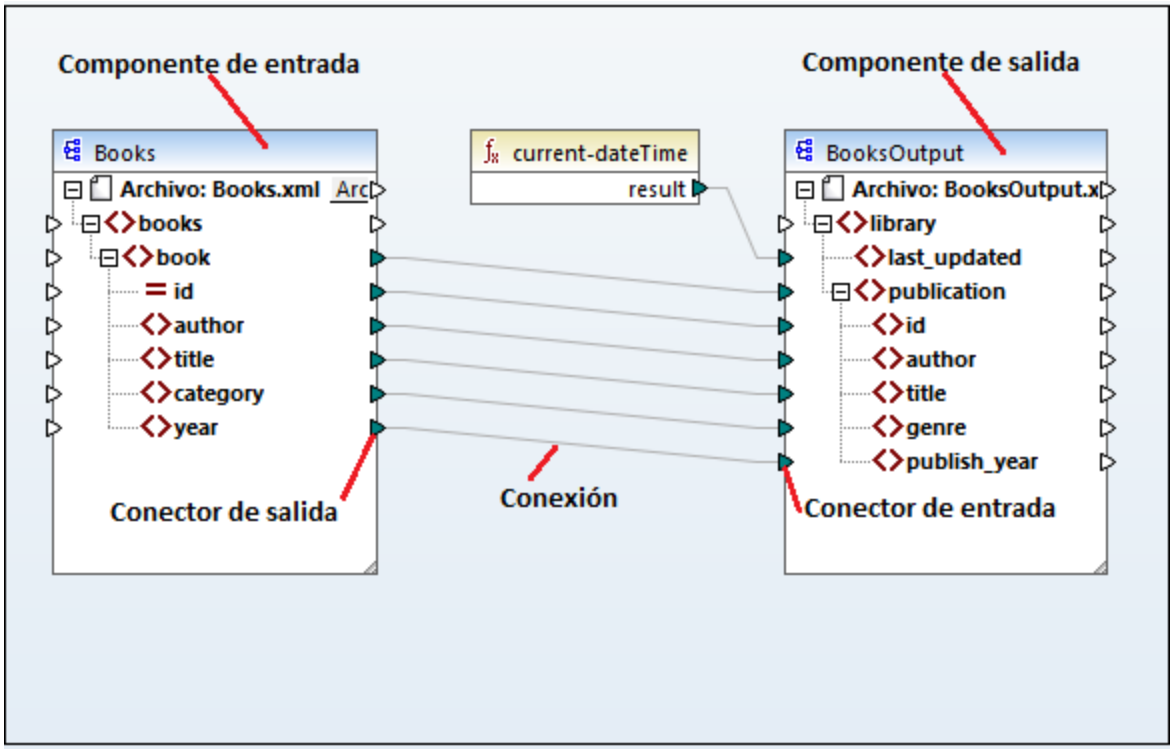
Panel Resultados de StyleVision (ediciones Enterprise y Professional)

Si tiene instalado Altova StyleVision podrá ver los paneles de resultados de StyleVision junto al panel **Resultados**. Estos paneles incluyen botones para obtener una vista previa de los resultados de StyleVision y guardarlos en formato HTML, RTF, PDF y Word 2007+. Esto es posible gracias a los archivos SPS (hojas de estilo StyleVision Power) diseñados en StyleVision y asignados a un componente de asignación en MapForce.

2 Fundamentos de la asignación de datos

Un diseño de asignación de datos (o simplemente asignación de datos) es la representación visual de cómo se deben transformar datos de un formato en otro. Una asignación consiste en *componentes* que se añaden al área de asignación para crear transformaciones de datos. Una asignación válida está formada por uno o varios *componentes de origen* que están conectados a uno o varios *componentes de destino*. La asignación se puede ejecutar y MapForce ofrece una vista previa del resultado. También puede generar código desde MapForce y ejecutar la asignación en una aplicación externa. Por último, puede compilar la asignación en un archivo de ejecución de MapForce y automatizar la ejecución con MapForce Server o FlowForce Server. MapForce guarda las asignaciones en archivos `.mfd`.

En la imagen siguiente puede ver la estructura básica de una asignación de datos:



Asignación nueva

Para crear una asignación nueva haga clic en el botón (**Nuevo**) de la barra de herramientas. También puede hacer clic en el menú **Archivo** y después en **Nuevo**. El paso siguiente es agregar componentes 36 a la asignación y trazar las conexiones 46 correspondientes.

Partes de una asignación

En los apartados siguientes describimos las partes principales de una asignación.

Componente

En MapForce un *componente* es el elemento gráfico que representa visualmente la estructura de los datos o el elemento que determina cómo se deben transformar los datos. Los componentes son piezas fundamentales

necesarias para construir una asignación y aparecen en el área de asignación de MapForce en forma de rectángulo. Los componentes pueden dividirse en dos grandes grupos:

- Componentes de origen y de destino
- Componentes estructurales 112 y de transformación 146

Estos dos grupos no se excluyen mutuamente. El primer grupo refleja las relaciones entre componentes, es decir, un componente puede ser de origen para un componente y de destino para otro. MapForce lee los datos de los componentes de origen y escribe daos en los componentes de destino. Al ejecutarse la asignación, el componente de destino da a MapForce instrucciones para generar archivos de salida o generar el resultado como valor de cadena para poder procesarlo en una aplicación externa. A continuación describimos los tipos de componentes del primer grupo:

- El componente de *origen* se encuentra a la izquierda del componente de destino. En este componente es donde MapForce lee los datos.
- El componente de *destino* está a la derecha del de origen. Aquí es donde MapForce escribe los datos.
- Los componentes de *paso a través* son un subtipo de los componentes de origen y destino y actúan tanto como componentes de origen como de destino. Para más información consulte Asignaciones encadenadas 94. Solamente pueden ser componentes de paso a través los componentes estructurales.

En el segundo grupo (componentes estructurales y de transformación)se puede ver si un componente tiene una estructura de datos o si se usa para transformar los datos asignados desde otro componente.

Para saber más sobre componentes y acciones relacionadas con los componentes consulte Componentes 32 .

Conector
Un conector es un pequeño triángulo situado a la izquierda o derecha de un componente. Los conectores de entrada están situados a la izquierda del componente y muestran puntos de entrada *para ese componente*. Los conectores de salida están situados a la derecha del componente y muestran puntos de salida *para ese componente*.

Conexión
Una conexión es la línea que se puede dibujar para unir dos conectores. Al dibujar la línea y crear la conexión, estamos dando a MapForce instrucciones de transformar lo datos de una forma determinada, como por ejemplo leer datos de un documento XML y escribirlos en otro documento XML.

Temas de esta sección

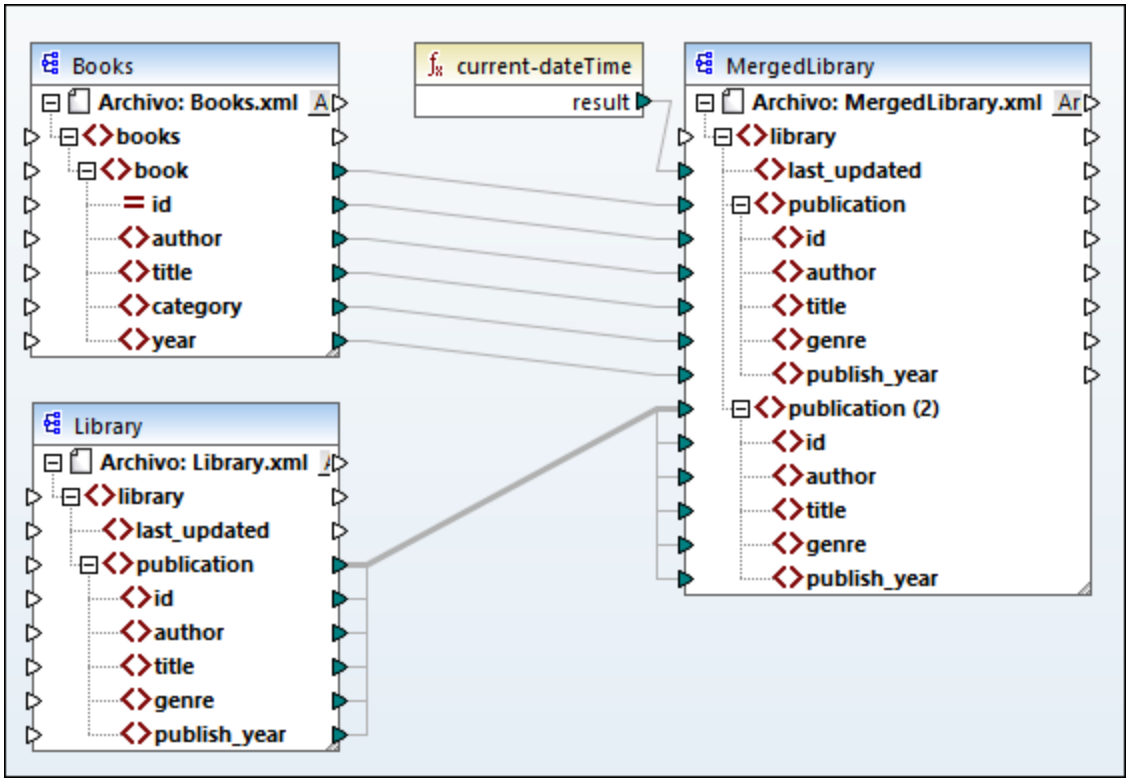
En esta sección explicamos las tareas y los conceptos más comunes de MapForce. Hemos dividido la sección en varios apartados:

- Componentes 32
- Conexiones 46
- Procedimientos y funciones generales 64

2.1 Componentes

Los componentes son los elementos centrales de los diseños de asignación de datos en MapForce. En el área de asignación de MapForce los componentes aparecen en forma de rectángulo. En este apartado encontrará un resumen de los componentes estructurales y de transformación (*véase el ejemplo siguiente*). La distinción se basa en si un componente tiene una estructura de datos o si se usa para transformar datos. Consulte la descripción de estos tipos en las subsecciones siguientes. Consulte también Tareas generales 30 . Además de componentes estructurales y de transformación, también puede añadir comentarios a las asignaciones (*véase Comentarios más abajo*).

Ejemplo de los componentes
En la asignación siguiente se pueden ver dos componentes de datos de origen (`Books` y `Library`), un componente de datos de destino (`MergedLibrary`) y un componente de transformación (la función **`current-dateTime`**).



Componentes estructurales

Los componentes estructurales representan la estructura abstracta de unos datos (por ejemplo, un archivo XML). Puede encontrar la lista de componentes estructurales que se pueden usar como orígenes y destinos de datos en Componentes estructurales 112. Los componentes estructurales pueden leer datos de ciertos orígenes de datos, escribir datos en otros y almacenar datos en alguna fase intermedia del proceso de asignación (por ejemplo, para previsualizar datos). En la tabla siguiente aparece una vista general de los componentes estructurales y los botones de la barra de herramientas correspondientes.

Icono	Descripción
	Componente XML
	Componente de texto (*ediciones Professional y Enterprise*)
	Componente de base de datos (*ediciones Professional y Enterprise* para bases de datos SQL; *ediciones Professional y Enterprise* para bases de datos NoSQL)
	Componente JSON (*edición Enterprise*)
	Componente de Microsoft Excel (*edición Enterprise*)
	Componente EDI (*edición Enterprise*)
	Componente XBRL (*edición Enterprise*)
	Protocol Buffers (*edición Enterprise*)

Componentes de transformación

Con los componentes de transformación puede transformar datos[196], almacenar resultados intermedios de una asignación[160] para procesarlos más adelante, reemplazar un valor por otro[184], ordenar[172], agrupar[279] y filtrar[178] datos. En la tabla siguiente aparece una vista general de los componentes de transformación y los botones de la barra de herramientas correspondientes.

Icono	Descripción
	Entrada simple
	Salida simple
	Componente de filtro
	Componente de ordenación
	Función integrada
	Función definida por el usuario
	Componente WHERE/ORDER de SQL/NoSQL (*ediciones Professional y Enterprise*)
	Componente de asignación de valores
	Variable
	Función de servicio web (*edición Enterprise*)
	Excepción (*ediciones Professional y Enterprise*)

Icono	Descripción
	Constante
	Condición If-Else
	Componente de combinación (*ediciones Professional y Enterprise*)

Comentarios

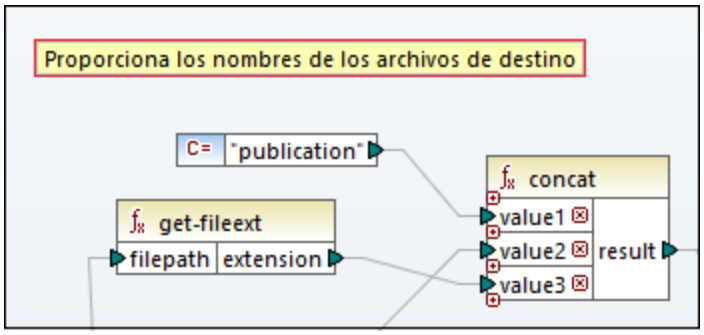
MapForce permite añadir comentarios como componentes independientes y también como notas bajo componentes que ya existan.

Comentarios de componentes

Los comentarios de los componentes son cajas autónomas en las que se visualizan varias líneas de texto y que no se pueden conectar a ningún otro componente. Para añadir un comentario a un componente tiene varias opciones:

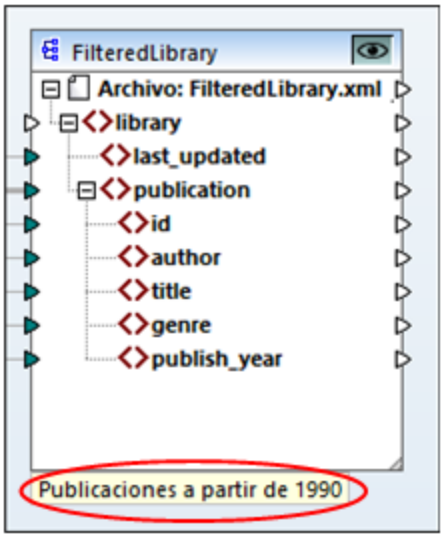
- Seleccione el comando  de la barra de herramientas, que abre un cuadro de diálogo donde puede escribir el comentario.
- Seleccione el comando de menú **Insertar | Comentario**, que abre un cuadro de diálogo donde puede escribir el comentario.
- Haga doble clic en un área vacía de la asignación, escriba el símbolo #, escriba un comentario y pulse la tecla **Entrar**. El símbolo # no aparece en la caja del comentario.

Para mover un comentario, arrástrelo a la ubicación deseada. Para eliminar un comentario, haga clic en su caja y pulse la tecla **Suprimir**. Más abajo puede ver un ejemplo de un comentario de componente (*rectángulo rojo*).



Comentarios de componentes

Además de los comentarios autónomos, también puede añadir comentarios a componentes que ya existan. Estos comentarios aparecen bajo el componente (*óvalo rojo*).



Para añadir un comentario a un componente que ya exista también tiene varias opciones:

- Haga clic con el botón derecho dentro del componente y seleccione **Editar comentario** en el menú contextual. Se abre un cuadro de diálogo en el que puede introducir un comentario.
- Seleccione el componente al que quiere añadir el comentario. Después seleccione **Editar comentario** en el menú **Componente**. Se abre un cuadro de diálogo en el que puede introducir un comentario.

Puede limitar los comentarios de componentes a un número concreto de líneas, que puede definir en el menú **Herramientas | Opciones | General | Visualización de asignaciones**. Para más información consulte Opciones 472.

Para eliminar un comentario de un componente:

- Haga doble clic en el comentario, borre todo el texto y pulse la tecla **Entrar**.
- Haga clic con el botón derecho en el comentario o dentro del componente, seleccione **Editar comentario** del menú contextual, borre el texto y haga clic en **Aceptar**.

Editar comentarios
Puede editar los dos tipos de comentarios de varias formas:

- Haga doble clic en el texto del componente y edite el texto directamente en la caja. Pulse la tecla **Entrar**.
- Haga clic con el botón derecho en la caja del comentario, edite el texto en el cuadro de diálogo **Editar comentario** y haga clic en **Aceptar**. También puede acceder al cuadro de diálogo **Editar comentarios** haciendo clic con el botón derecho dentro del componente y seleccionando la opción **Editar comentario** en el menú contextual.

En esta sección

Esta sección sobre componentes se divide en distintos apartados:

2.1.1 Agregar componentes

En este apartado explicamos cómo agregar componentes a una asignación. Para agregar un componente primero debe crear un diseño de asignación nuevo [30] y después seguir estos pasos:

- En el menú **Insertar**, elija un tipo de componente (p.ej. **Esquema o archivo XML**).
- Arrastre un archivo desde el explorador de Windows hasta el área de asignación.
- Haga clic en el botón correspondiente de la barra de herramientas **Insertar componente** (*imagen siguiente*).

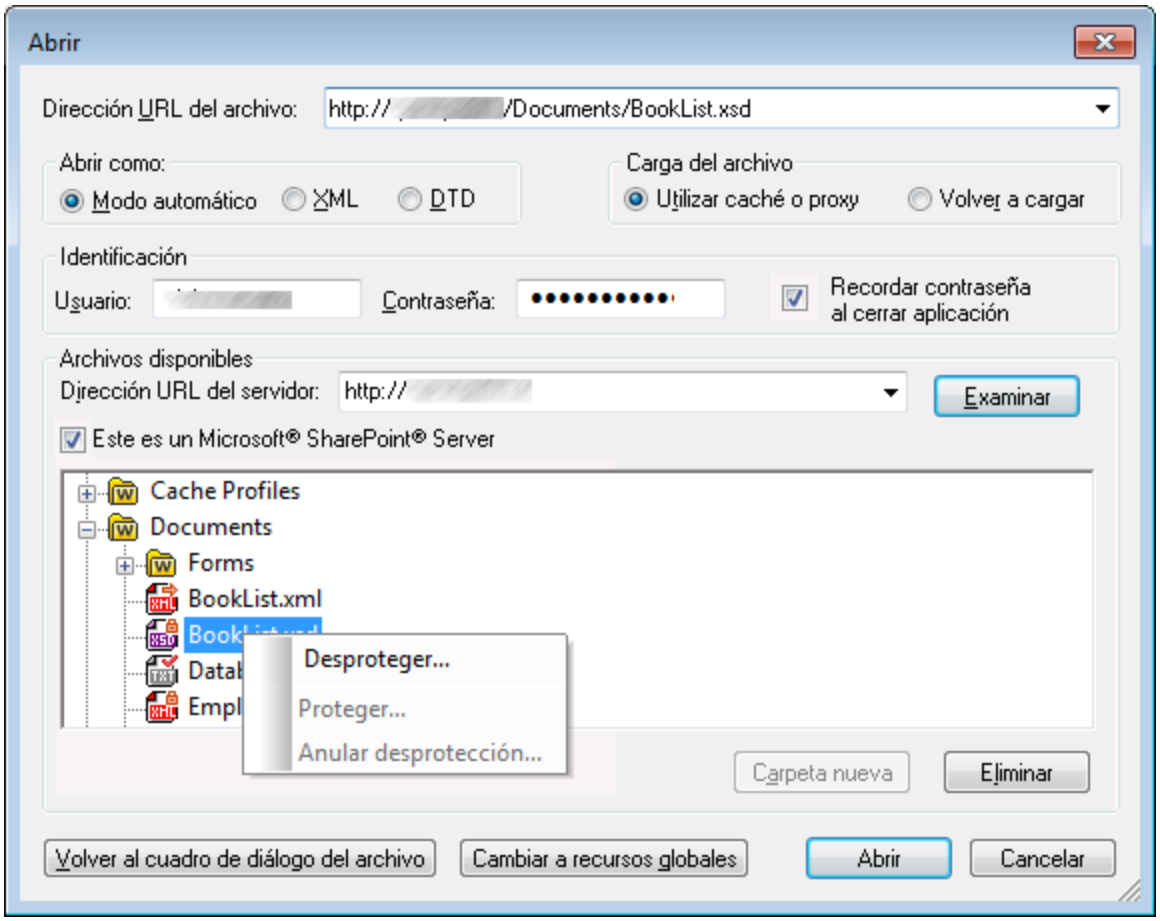
Cada tipo de componente tiene un objetivo y comportamiento concretos Para ver un resumen de los componentes consulte Componentes [32]. Si quiere saber más acerca de las estructuras de datos que se pueden usar como componentes de origen and de destino en consulte Orígenes y destinos de datos [112]. Para más información sobre los componentes de MapForce que se usan para almacenar datos temporalmente o para transformarlos consulte Diseño de asignaciones [146].

Si quiere agregar un componente estructural a una asignación, puede elegir entre agregar un archivo local, un componente desde una URL o uno de la lista de recursos globales (*véanse las subsecciones siguientes*).

Agregar componentes desde una URL

Sólo se pueden agregar componentes desde una URL a componente de origen [30]. Los protocolos compatibles son HTTP, HTTPS y FTP. En función del tipo de estructura de datos, las instrucciones sobre cómo añadir un componente desde una URL pueden variar. Para la mayoría de las estructuras de datos sirven estas instrucciones:

1. Seleccione componente que quiere agregar (p.ej. **Archivo o esquema XML**).
2. Haga clic en **Cambiar a URL** en el cuadro de diálogo **Abrir**.
3. Introduzca la URL en la caja de texto *Dirección URL del archivo* y haga clic en **Abrir** (*imagen siguiente*).



En la lista siguiente se describen las opciones del cuadro de diálogo **Opciones**.

- *Abrir como:* Esta opción define la gramática del analizador. La opción predeterminada y recomendada es *Automático*.

- *Cargar archivos:* Si sabe que no es probable que el archivo que va a cargar cambie, seleccione *Utilizar caché o proxy* para guardar los datos en la memoria caché y acelerar el proceso de carga. Seleccione *Volver a cargar* para que el archivo se cargue de nuevo cada vez que abra la asignación.

- *Identificación:* Si el servidor requiere autenticación con contraseña le pedirá que indique un nombre de usuario y una contraseña. Si quiere que MapForce recuerde sus datos de acceso la próxima vez que lo inicie, introduzca su nombre de usuario y contraseña en el cuadro de diálogo Abrir y marque la casilla *Recordar contraseña al cerrar aplicación*.

- *Dirección URL del servidor:* En el caso de los servidores compatibles con WebDAV (Autoría y versionado distribuidos por Web) puede explorar archivos una vez haya introducido la URL del servidor en la caja de texto *URL del servidor* y hecho clic en **Examinar**. Aunque en la vista previa aparecen todos los tipos de archivos, debe asegurarse de que abre el mismo tipo de archivo que el que en el paso 1 más arriba. De lo contrario ocurrirán errores.

- *Desproteger/Proteger:* Si está usando un servidor Microsoft SharePoint Server, marque la casilla que así lo indica. Así podrá acceder a una vista previa del estado del archivo. Si quiere asegurarse de que nadie más puede editar el archivo en el servidor mientras lo está usando, haga clic con el botón

derecho en el archivo y seleccione **Proteger** (*imagen anterior*). Para desproteger archivos protegidos haga clic con el botón derecho en el archivo y seleccione **Desproteger**.

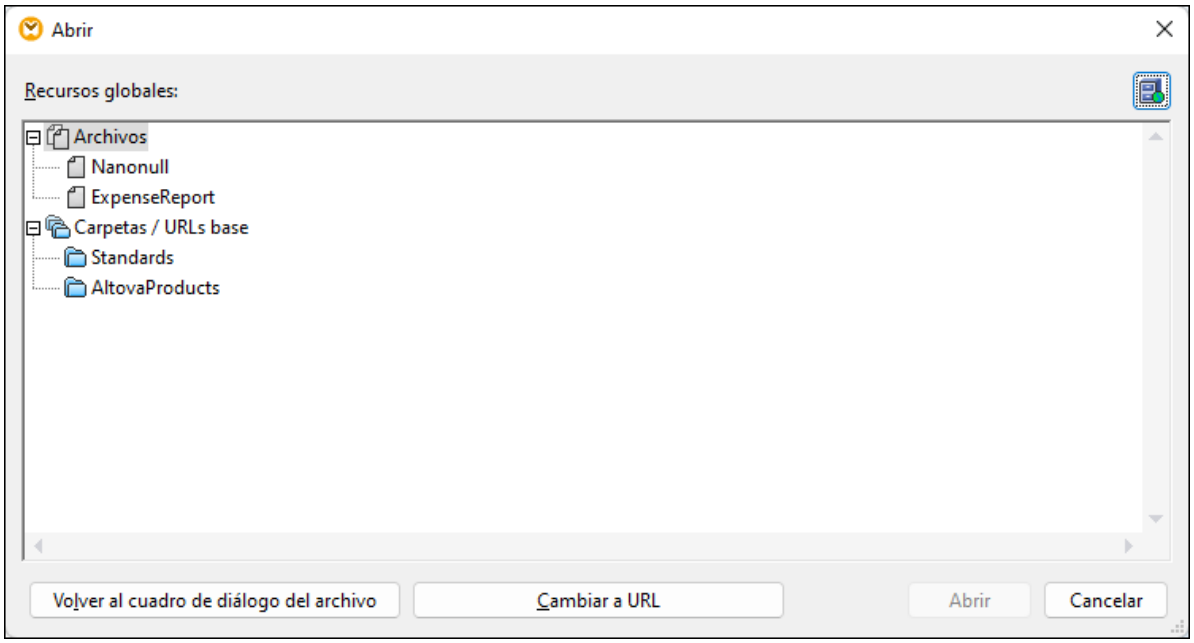
- *Cambiar al cuadro de diálogo Archivo:* Haga clic en este botón para ir al cuadro de diálogo en el que puede seleccionar un archivo local.

- *Cambiar a recursos globales:* Haga clic en este botón para ir al cuadro de diálogo en el que puede seleccionar un recurso global.

Agregar recursos globales

Puede añadir a la asignación cualquier base de datos (*ediciones Professional y Enterprise*), archivo o carpeta que haya definido como recurso global. Para más información, consulte Recursos globales de Altova[432]. En función del tipo de estructura de datos con el que esté trabajando, las instrucciones sobre cómo añadir un componente desde una URL pueden variar. Para la mayoría de las estructuras de datos sirven estas instrucciones:

1. Seleccione el componente que quiere agregar (p.ej. **Archivo o esquema XML**).
2. Haga clic en **Cambiar a URL** en el cuadro de diálogo **Abrir**.
3. Seleccione uno de los recursos de la lista y haga clic en **Abrir** (*imagen siguiente*).



Si quiere agregar, editar y eliminar recursos globales haga clic en el icono **Administrar recursos globales** (*en un círculo rojo*). El cuadro de diálogo **Abrir** para recursos globales permite volver a cambiar al cuadro de diálogo de archivos locales (**Cambiar al cuadro de diálogo Archivo**) o abrir un archivo desde una URL (**Cambiar a URL**).

2.1.2 Componentes básicos

En este apartado explicamos cómo configurar, buscar y manipular componentes estructurales[32]. Para más información siga leyendo.

Cambiar la configuración de los componentes

Tras añadir un componente al área de asignación podrá configurarlo desde el cuadro de diálogo **Configuración del componente**. Puede abrir este cuadro de diálogo de una de las siguientes maneras:

- Haga doble clic en el encabezado del componente.
- Seleccione el componente y haga clic en **Propiedades** en el menú **Componente**.
- Haga clic con el botón derecho en el encabezado del componente y luego en **Propiedades**.

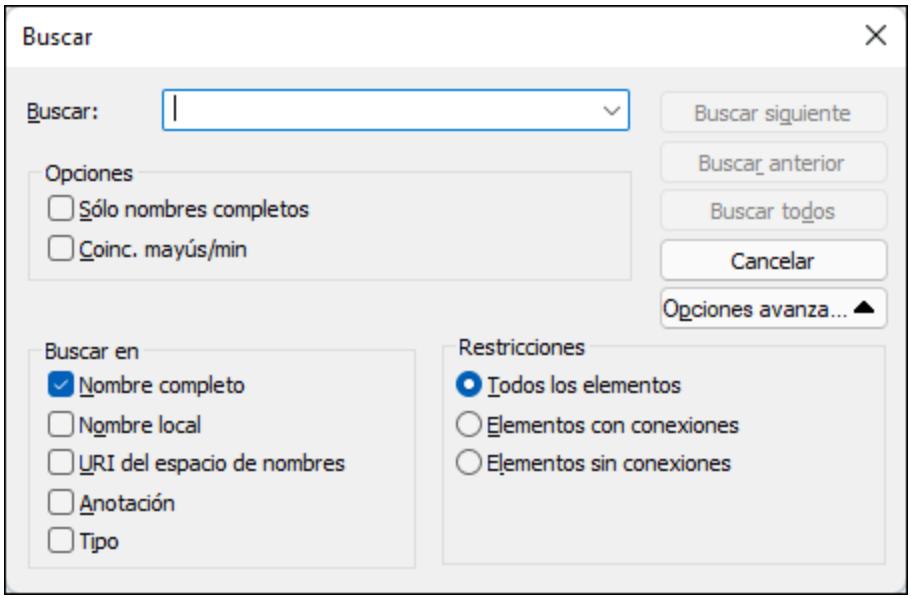
Puede ver la lista de opciones de configuración disponibles en Configuración de componentes XML[114].

En los componentes basados en archivos (p. ej. un archivo XML) los botones Archivo (*Basic Edition*) o Archivo/Cadena (*ediciones Professional y Enterprise*) aparecen junto al nodo raíz. Con este botón puede acceder a las opciones avanzadas, con las que puede procesar o generar varios archivos en una sola asignación. Para más información consulte la sección Procesar varios archivos de entrada o salida[402].

Buscar dentro de los componentes

Para buscar un nodo en concreto dentro de un componente siga estos pasos:

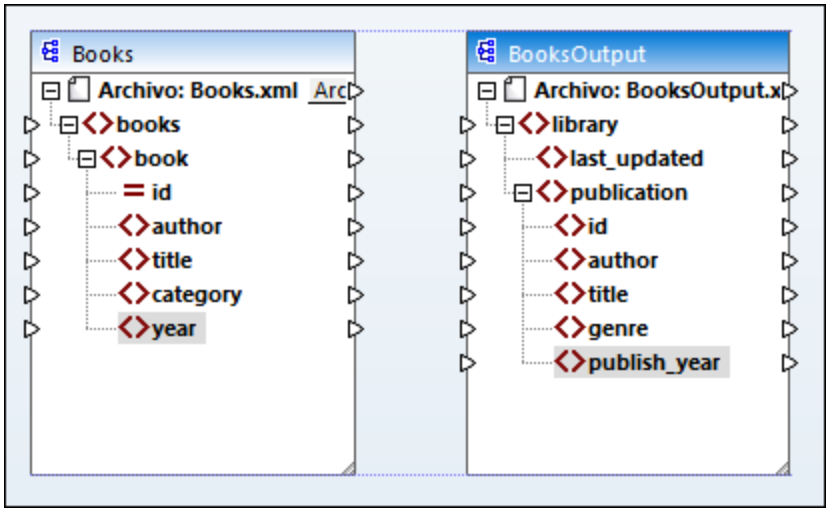
1. Haga clic en el componente donde quiere buscar y pulse **Ctrl+F**.
2. Introduzca un término de búsqueda y haga clic en Buscar siguiente/anterior/todos (*imagen siguiente*).



Use las opciones **Avanzadas** para definir qué elementos (nodos) quiere buscar. También puede restringir las opciones de búsqueda en función de conexiones específicas.

Alinear componentes

Al mover componentes en el área de asignación, MapForce muestra unas líneas guía (líneas de puntos) que ayudan a alinear componentes (*imagen siguiente*).



Para habilitar esta opción, siga estos pasos:

1. En el menú **Herramientas** haga clic en el comando **Opciones**.
2. En la pestaña de opciones **Edición** active/desactive la casilla **Alinear componentes al arrastrarlos con el ratón**.

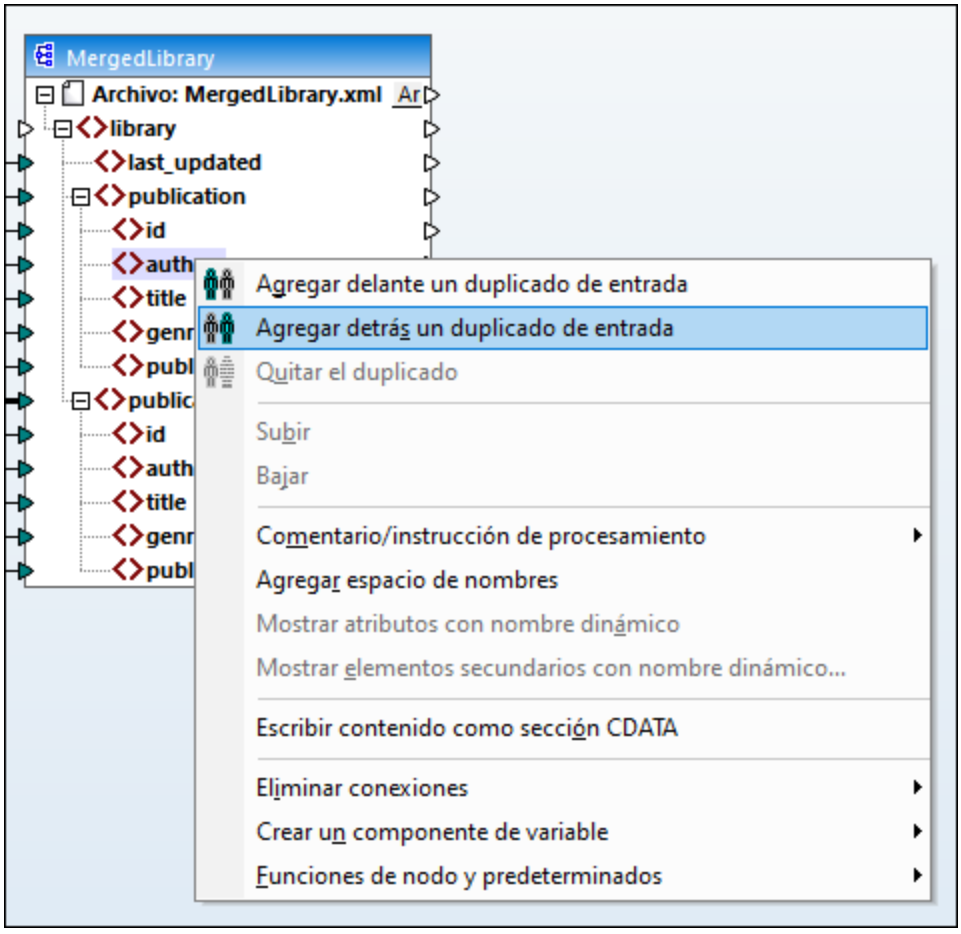
Duplicar elementos de entrada

A veces es necesario configurar un componente para que acepte datos de varios orígenes de datos a la vez. Si quiere que el esquema de destino acepte datos de más de un esquema de origen, puede duplicar los nodos de entrada en el componente de destino. Si lo hace, debe tener en cuenta que: Los nodos duplicados aceptan datos, pero no se pueden asignar datos de nodos duplicados. Puede duplicar tantos nodos como quiera.

Hay dos maneras de duplicar entradas: (i) seleccionar **Agregar delante/detrás un duplicado de entrada** en el menú contextual y (ii) conectar un nodo de origen con un nodo de destino que ya esté conectado a un nodo distinto. A continuación describimos la primera opción. Puede encontrar información sobre la segunda en el segundo tutorial 89.

Agregar duplicados de entrada

Para crear un duplicado de un elemento de entrada, haga clic con el botón derecho en el elemento y seleccione **Agregar delante/detrás un duplicado de entrada** en el menú contextual (*imagen siguiente*). En la imagen siguiente se va a duplicar el nodo `author` para poder asignar los datos al nodo duplicado desde otro nodo de origen.



Nota: no está permitido crear duplicados de atributos XML porque daría lugar a una instancia XML no válida.

2.1.3 Rutas de acceso de archivos

Un archivo de diseño de asignación de datos (***.mfd**) puede tener referencias a varios archivos de instancia o de esquema. Los archivos de esquema le sirven a MapForce para determinar la estructura de los datos que se deben asignar y validarlos. En las ediciones MapForce Professional y Enterprise las asignaciones también pueden incluir referencias a archivos StyleVision Power Stylesheet (***.sps**), que se usan para dar formato a los datos de salida como PDF, HTML o Word. Las asignaciones también pueden contener referencias a bases de datos basadas en archivos, como Microsoft Access o SQLite.

Cuando añadimos un componente a la asignación, MapForce crea referencias a los archivos. No obstante, estas referencias se pueden cambiar y configurar a mano en cualquier momento.

En esta sección explicamos cómo configurar y modificar las rutas de acceso de los diferentes tipos de archivo a los que se hace referencia en las asignaciones de datos. Estos son los apartados de esta sección:

2.1.3.1 Rutas de acceso absolutas y relativas

En este apartado explicamos cómo usar las rutas de acceso absolutas y relativas de los archivos a los que hace referencia un componente. Una ruta absoluta muestra la ubicación completa de un archivo y empieza por el directorio raíz. En la imagen siguiente puede ver las rutas absolutas dentro del recuadro rojo. En una ruta relativa se puede ver la ubicación del archivo de manera relativa al directorio de trabajo actual: p.ej. **`Books.xml`**.

En el cuadro de diálogo **Configuración del componente** (*ejemplo siguiente*) puede indicar rutas absolutas o relativas para los distintos archivos a los que hace referencia el componente. A continuación puede ver una lista de esos archivos:

- Archivos de entrada de los que MapForce lee datos;
- Archivos de salida en los que MapForce escribe datos;
- Archivos de esquema, que se pueden aplicar a componentes que tienen un esquema;
- Archivos de estructura, que se usan como parámetros de entrada o salida para funciones definidas por el usuario y variables;
- Archivos StyleVision Power Stylesheet (**`*.sps`**), que se usan para dar formato a los datos para formatos de salida como PDF, HTML y Word;
- Archivos de BD en el caso de los componentes de BD (*en las ediciones Professional y Enterprise*).

Rutas relativas y las acciones cortar y pegar
Si copia un componente de una asignación y lo pega en otra, MapForce comprueba si las rutas relativas de los archivos de esquema se pueden resolver con respecto a la carpeta de la asignación de destino. Si esas rutas no se pueden resolver, la aplicación le pedirá que las convierta en absolutas.

Rutas de acceso rotas
Cuando añada o modifique una referencia de archivo en una asignación y no se pueda resolver la ruta de acceso, MapForce emite un mensaje de advertencia. No obstante, pueden darse referencias de ruta de acceso rotas en estos casos:

- Cuando use rutas de acceso relativas y después mueva el archivo de asignación a un directorio nuevo sin mover también los archivos de esquema y de instancia.
- Cuando use rutas de acceso absolutas de archivos que están en el mismo directorio que el archivo de asignación y después mueva el directorio a otra ubicación.

Cuando esto ocurra, MapForce resaltará el componente en color rojo. En casos así, la solución consiste en hacer doble clic en el título del componente y actualizar las referencias de ruta de acceso rotas en el cuadro de diálogo **Configuración del componente**. Consulte también Cambiar configuración de los componentes 39
.

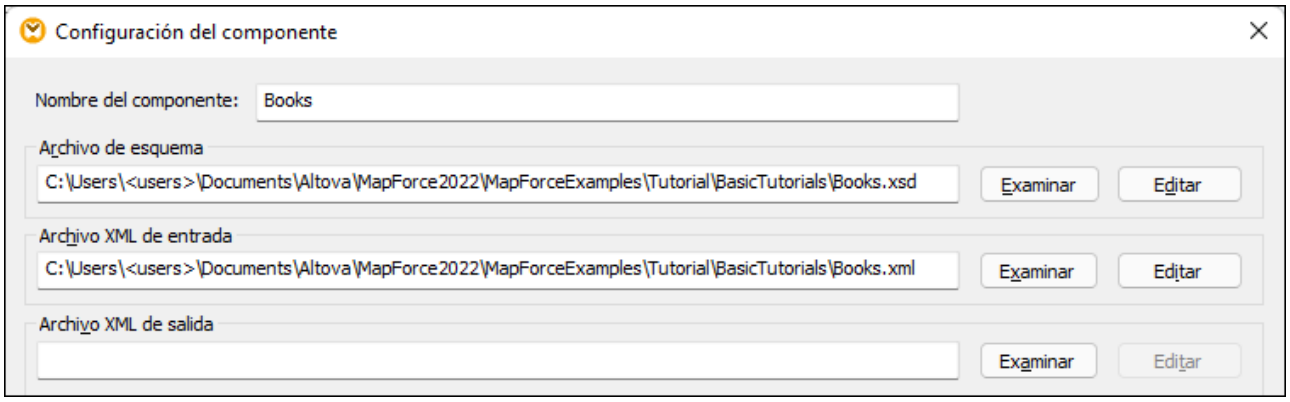
Ejemplo: Componente XML

En el ejemplo siguiente se ve cómo se pueden usar las rutas de acceso en un componente XML. Si quiere guardar todos los archivos de la asignación como relativos al archivo de asignación (**`.mfd`**) debe marcar la casilla **Guardar todas las rutas de acceso de archivos como relativas al archivo MFD** de la parte inferior del cuadro de diálogo **Configuración del componente**. Esta es la opción predeterminada y recomendada, y afecta a todos los archivos a los que hace referencia el componente (*en el recuadro rojo en la imagen siguiente*). Si todavía no ha guardado la asignación, en el cuadro de diálogo **Configuración del componente**

verá que las rutas de acceso al esquema y a los archivos de instancia son absolutas. Para convertirlas en relativas:

1. Cree una asignación nueva[30] y agregue un componente estructural[36]: p. ej. un archivo XML que tenga asignado un esquema XML.
2. Haga doble clic el encabezado del componente para abrir el cuadro de diálogo **Configuración del componente**.
3. Marque la casilla **Guardar todas las rutas de acceso de archivos como relativas al archivo MFD** de la parte inferior del cuadro de diálogo **Configuración del componente**.
4. Guarde la asignación.
5. Ahora puede volver a abrir el cuadro de diálogo **Configuración del componente**, que ahora contendrá rutas relativas en los campos de texto correspondientes.

Nota: Las rutas que hacen referencia a unidades no locales o usan una URL no se pueden convertir en relativas.



Si se marca la casilla **Guardar todas las rutas de acceso de archivos como relativas al archivo MFD** MapForce hará un seguimiento de los archivos a los que hace referencia el componente incluso aunque guarde la asignación en una carpeta distinta. Si todos los archivos están en el mismo directorio que la asignación, las referencias de las rutas no se rompen aunque mueva todo el directorio a otra ubicación en disco.

La opción **Guardar todas las rutas de acceso de archivos** como relativas al archivo MFD afecta a estos archivos:

- Archivos de estructura, que usan parámetros complejos de entrada o salida para funciones definidas por el usuario, y variables de tipo complejo;
- Archivos planos de entrada o salida (*en las ediciones Professional y Enterprise*);
- Archivos de esquema a los que hacen referencia los componentes de BD que admiten campos XML (*en las ediciones Professional y Enterprise*);
- Archivos de BD (*en las ediciones Professional y Enterprise*);
- Archivos XBRL, FlexText, EDI, Excel 2007+ o JSON de entrada o salida (*en la edición Enterprise*);

Nota: Cuando genere el código de programa, compile archivos de ejecución de MapForce Server (**`.mfx`**) o implemente la asignación en FlowForce Server, las rutas relativas se convierten en absolutas si marca la casilla **Convertir las rutas de acceso en absolutas en el código generado** en las opciones de la asignación[76]. Para más información consulte Rutas de acceso según el entorno de ejecución[44].

2.1.3.2 Rutas de acceso según el entorno de ejecución

Si quiere generar código a partir de asignaciones, los archivos que se generan los ejecuta el entorno de destino que haya elegido: por ejemplo,RaptorXML Server . Para que la asignación se pueda ejecutar correctamente las rutas relativas deben corresponderse con el formato del entorno de ejecución. Estas son las rutas base para los distintos lenguajes de destino:

Lenguaje de destino	Ruta base
XSLT, XSLT2, XSLT3	Ruta del archivo XSLT.
XQuery*	Ruta del archivo XQuery.
C++, C#, Java*	Directorio de trabajo de la aplicación generada.
Motor integrado* (para la vista previa de la asignación en con MapForce)	Ruta del archivo de asignación (**.mfd**).
Motor integrado* (para ejecutar la asignación con MapForce Server)	El directorio de trabajo actual.
Motor integrado* (para ejecutar la asignación con MapForce Windows Server bajo el Server Archivos)	El directorio de trabajo del trabajo o el de FlowForce Server.

** Lenguajes disponibles en las ediciones MapForce Professional y Enterprise.*

De ruta relativa a ruta absoluta
Cuando genere el código de programa, compile archivos de ejecución de MapForce Server (**.mfx**) o implemente la asignación en FlowForce Server, las rutas relativas se convierten en absolutas si marca la casilla **Convertir las rutas de acceso en absolutas en el código generado** en las opciones de la asignación 76.

Si genera código y esa casilla está marcada, MapForce resuelve las rutas relativas basadas en el directorio del archivo **.mfd** y las convierte en absolutas en el código generado. Esta opción afecta a las rutas de los siguientes archivos:

- Archivos de instancia de entrada y de salida de todos los componentes basados en archivos;
- Archivos de BD Access y SQLite que se usen como componentes de asignación (*en las ediciones Professional y Enterprise*).

Rutas de acceso a bibliotecas en el código generado

Los archivos de asignación pueden contener referencias de rutas de acceso a distintas bibliotecas, Por ejemplo,, puede importar funciones definidas por el usuario desde otro archivo de asignación, desde bibliotecas personales XSLT, XQuery*, C#* o Java* o desde archivos **.mff*** (MapForce Funcion). Para más información consulte Gestionar bibliotecas de funciones 200.

** Características disponibles en las ediciones MapForce Professional y Enterprise.*

La opción **Convertir las rutas de acceso en absolutas** en el código generado solamente afecta a los componentes de asignación, pero no a las rutas de bibliotecas externas. En el caso de las bibliotecas que no sean XSLT ni XQuery, la ruta de acceso se convierte en absoluta en el código generado. Por ejemplo, si un

archivo de asignación contiene referencias de biblioteca como archivos .NET **.dll** o archivos Java **.class**, y si quiere ejecutar el código generado en algún otro entorno, las bibliotecas a las que se hace referencia deben existir en la misma ruta que el entorno de destino.

Si quiere generar un archivo XSLT o XQuery a partir de una asignación, configure la ruta para que sea relativa al archivo XSLT o XQuery:

1. Abra la configuración de la asignación 76.
2. Marque la casilla **Bibliotecas de referencia con rutas relativas a los archivos XSLT / XQuery generados**. Asegúrese de que el archivo de biblioteca XSLT/XQuery existe en esa ruta.

2.2 Conexiones

Una conexión es una línea que conecta un elemento de origen con uno de destino. Las conexiones son una representación visual de cómo están asignados los datos de unos nodos a otros. Los apartados siguientes describen las distintas acciones relacionadas con conexiones que se pueden ejecutar.

Crear, copiar y eliminar conexiones

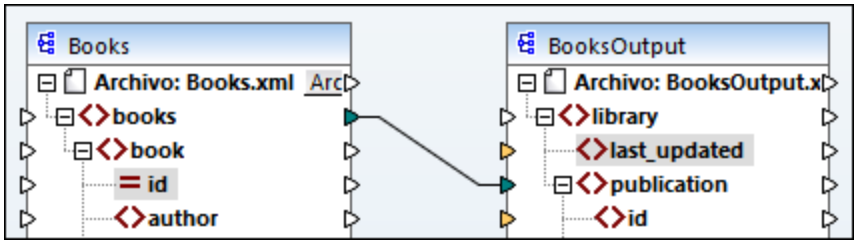
Para crear una conexión entre dos elementos pulse y mantenga pulsado el conector de salida 31 de un nodo de origen y arrástrelo hasta el nodo de destino. Un conector de entrada sólo acepta *una conexión*. Si intenta agregar una segunda conexión a un elemento de entrada, MapForce le pedirá que reemplace una conexión por otra o que duplique el elemento de entrada 40. Un elemento de salida sí que puede tener varias conexiones a distintos elementos de entrada.

Para copiar una conexión a otro elemento pulse y mantenga pulsada la parte gruesa que hay al final de la conexión (*véase la imagen de Mover conexiones*) y arrástrela hasta su destino mientras mantiene pulsada la tecla **Ctrl**.

Para eliminar una conexión haga clic en ella y pulsar la tecla **Supr**. También puede hacer clic con el botón derecho en la conexión y seleccionar **Eliminar** en el menú contextual.

Entradas obligatorias

Para facilitar la creación de asignaciones, MapForce resalta en color naranja las entradas obligatorias de los componentes de destino. En el ejemplo siguiente puede ver cómo al conectar el elemento `book` del componente `Books` al elemento `publication` del componente `BookOutput` se resaltan los conectores de los nodos obligatorios del componente `BookOutput`. Si no conecta las entradas obligatorias, los nodos correspondientes no se asignan a los elementos de destino y la asignación no será válida.



Conexiones antecesoras ausentes

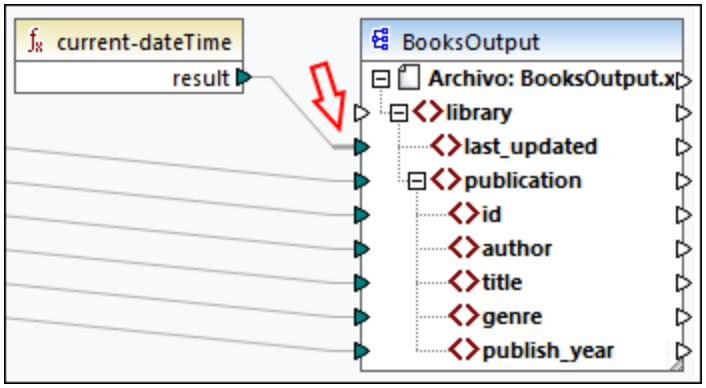
Cuando se crean conexiones entre nodos de origen y destino de forma manual, MapForce analiza los posibles resultados de la asignación. Si se crea una asignación entre dos nodos secundarios, MapForce puede emitir una notificación para sugerir una conexión entre el elemento primario del nodo de origen y el elemento primario del nodo de destino. Esta notificación le ayudará a evitar que en la ventana de vista previa de resultados aparezca un solo nodo secundario.

Si prefiere deshabilitar este tipo de notificaciones:

1. En el menú **Herramientas** haga clic en el comando **Opciones**.
2. Haga clic en el grupo **Mensajes** para abrirlo.
3. Desactive la casilla **Al crear una conexión nueva, sugerir conexión de elementos antecesores**.

Mover conexiones

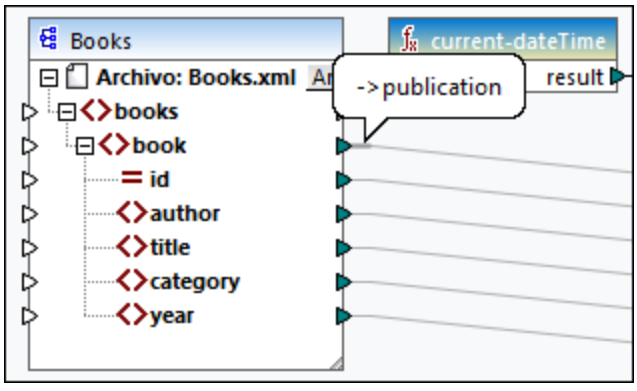
Para mover una conexión a otro nodo pulse y mantenga pulsada la parte gruesa que hay al final de la conexión (*véase la imagen de más abajo*) y arrástrela hasta su destino.



Ver la información rápida de las conexiones

La información rápida de las conexiones permite ver los nombres de los (i) nodos a los que se asignan datos o (ii) nodos desde los que se asignan datos. Para poder ver esa información pulse el botón de la barra de herramientas  (**Ver información rápida**). Para ver los nombres de los nodos a los que se asignan los datos haga que el cursor apunte a la parte gruesa de la conexión que se encuentra junto al conector de salida (*imagen siguiente*). Para ver el nombre de un nodo desde el que se asignan datos haga que el cursor apunte a la parte gruesa de la conexión que se encuentra junto al conector de entrada. Si se han definido varias conexiones desde un mismo elemento de salida, el número máximo de nombres que pueden aparecer en la información rápida es diez.

En el ejemplo siguiente el elemento de destino al que se asignan los datos del elemento `book` es `publication`.



Cambiar la configuración de las conexiones

Si quiere cambiar la configuración de las conexiones tiene varias opciones:

- Seleccionar una conexión Después haga clic en el comando **Propiedades** del menú **Conexión** .
- Haga doble clic en la conexión.
- Haga clic con el botón derecho en la conexión y después elija **Propiedades**.

Para más información consulte Configuración de las conexiones 56.

Resaltar conexiones de forma selectiva

MapForce permite resaltar conexiones en una asignación. Esta característica puede ser útil si una asignación tiene muchos componentes con muchas conexiones. Al resaltar conexiones de forma selectiva es más fácil comprobar si los nodos del componente seleccionado están asignados correctamente. Observe que el término *conector* de los botones de la barra de herramientas se refiere a una conexión en el sentido de una línea que une nodos de componente. A continuación puede ver las opciones disponibles.

	**Mostrar los conectores del componente seleccionado** (solo conexiones directas)
	**Mostrar los conectores desde su origen a su destino** (conexiones directas e indirectas)

Sólo conexiones directas

Si deja *sin pulsar* botón para **conexiones directas** puede ver todas las conexiones en negro. Si pulsa el botón para **conexiones directas** sólo aparecen en negro las conexiones relacionadas con el componente seleccionado en ese momento. El resto de conexiones aparecen de color gris pálido.

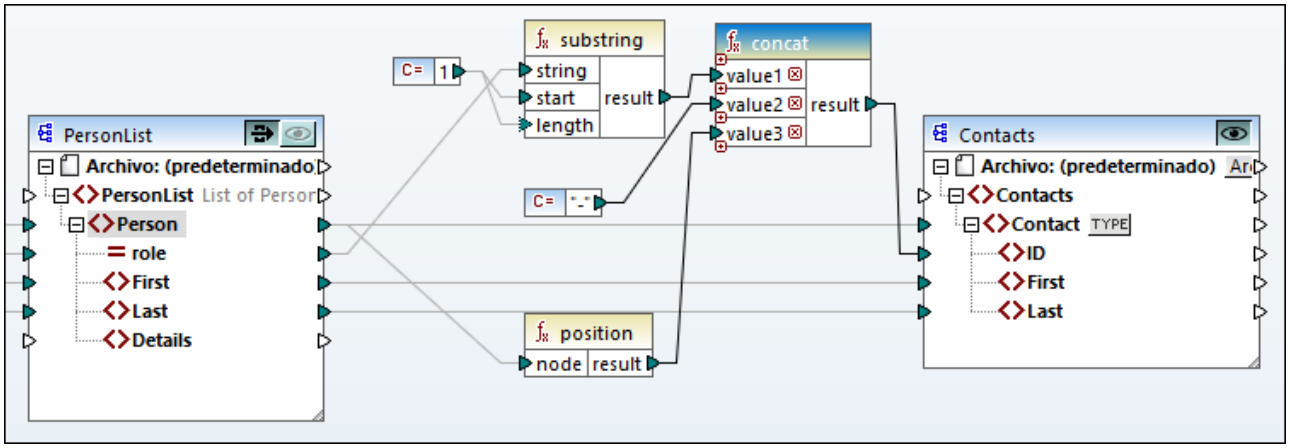
Conexiones directas e indirectas

El botón para **conexiones de origen a destino** se habilita solo si pulsa el botón para **conexiones directas**. Si pulsa el botón para **conexiones de origen a destino** puede hacer un seguimiento de las conexiones del componente seleccionado en ese momento, incluidas las conexiones directas y las de sus componentes conectados, hasta los archivos de origen y destino.

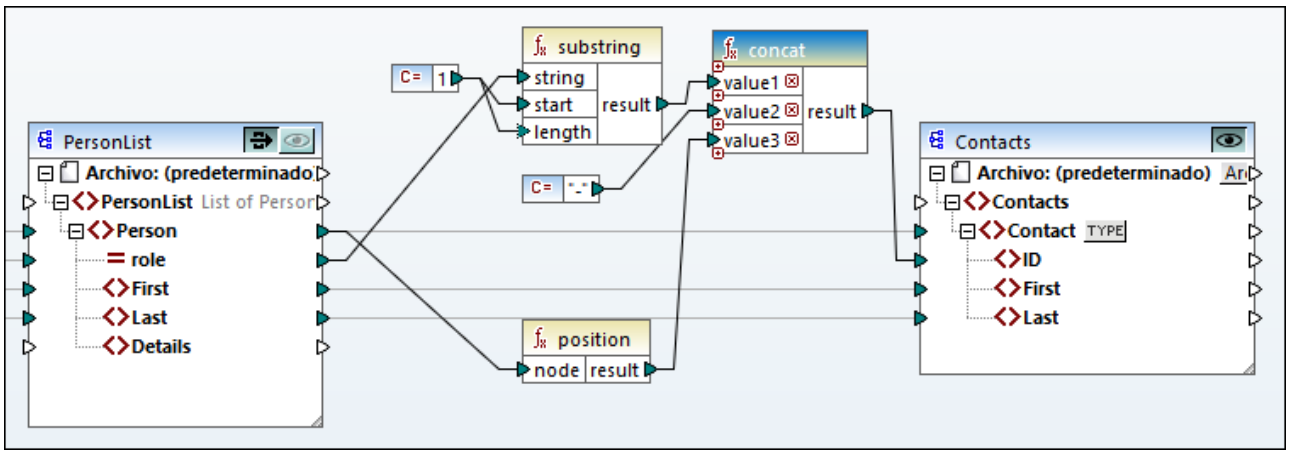
A continuación puede ver un ejemplo que explica cómo funcionan estas dos opciones.

Ejemplo

En la imagen siguiente puede ver una parte de la asignación **`ChainedPersonList.mfd`**, que puede encontrar en la carpeta **`MapForceExamples`**. En la asignación siguiente hemos pulsado el botón para **conexiones directas**, hemos hecho clic en el encabezado del componente **`concat`** pero todavía no hemos pulsado el botón Mostrar los conectores desde su origen a su destino. Esto quiere decir que ahora sólo las conexiones directas que van de la función **`concat`** a la constante **`"-"`**, los componentes `substring`, `position` y `Contacts`, aparecen en negro. El resto de conexiones de la asignación aparecen de color gris pálido.



El paso siguiente es pulsar el botón **Mostrar los conectores desde su origen a su destino**. En la imagen siguiente se pueden ver los cambios:



Al pulsar el botón **Mostrar los conectores desde su origen a su destino** hay más conexiones que aparecen en negro: (i) las conexiones que van de la función **`substring`** a la constante **`1`** y al componente `PersonList`, y (ii) la conexión que va de la función **`position`** al componente `PersonList`. El resto de conexiones entre el componente `PersonList` y su componente anterior siguen siendo de color gris pálido. Es decir, si pulsa el botón **Mostrar los conectores desde su origen a su destino** puede hacer un seguimiento de las conexiones directas de ese componente. Si el componente seleccionado está conectado a algún componente de transformación 33 (p.ej. funciones, constantes, filtros, componentes de ordenación, componentes SQL-NoSQL-WHERE/ORDER, condiciones if-else, asignaciones de valores), también podrá ver sus conexiones, lo que incluye los componentes estructurales 32 (como `PersonList`, más arriba) las variables, los componentes de combinación o las funciones de servicios web a las que están conectados estos componentes de transformación.

Apartados de esta sección

Esta sección sobre conexiones se divide en distintos apartados:

- Tipos de conexión 50
- Configuración de las conexiones 56

- Menú contextual de las conexiones 58
- Conexiones defectuosas 60
- Conservar conexiones tras eliminación de componentes 61

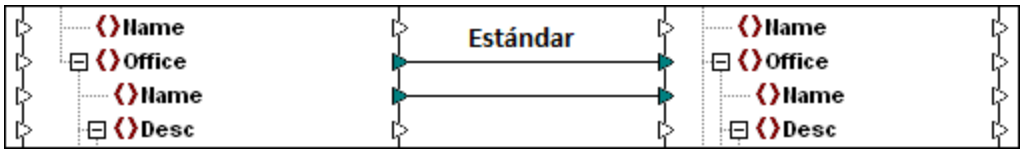
2.2.1 Tipos de conexión

Estos son los tipos de conexión que se pueden generar en MapForce:

- Conexiones basadas en el destino 50 (estándar)
- Conexiones basadas en el origen 50 (contenido mixto)
- Conectar los secundarios equivalentes 53
- Conexiones de copia total 55 (copiar los elementos secundarios)

Conexiones basadas en el destino vs. conexiones basadas en el origen

Las conexiones basadas en el destino y conexiones basadas en el origen se excluyen mutuamente. La elección de uno u otro tipo depende del orden en que se deban asignar los nodos. En las conexiones basadas en el destino, el orden de los nodos en el resultado viene dado por el esquema de *destino*. Este tipo de conexión se puede usar en la mayoría de asignaciones y es el tipo de conexión predeterminado en MapForce. Las conexiones basadas en destino aparecen como líneas sólidas (*imagen siguiente*).



Las conexiones basadas en destino pueden no ser adecuadas si quiere asignar nodos XML con contenido mixto (nodos secundarios y texto). En este caso recomendamos usar una conexión basada en el origen 50: El orden de los nodos en el resultado aquí viene dado por el esquema de *origen*.

Secundarios equivalentes y conexiones de copia total

Los secundarios equivalentes y conexiones de copia total pertenecen a un tipo distinto de conexiones. Estas conexiones asignan datos entre nodos con nodos secundarios que son parecidos o idénticos en los componentes de origen y destino. Las conexiones de copia total se parecen a las de secundarios equivalentes pero tienen solamente una conexión sólida en lugar de varias para no abarrotar la vista.

En esta sección explicamos los distintos tipo de conexión y en qué casos es útil cada uno.

2.2.1.1 Conexiones basadas en el origen

Una conexión basada en origen permite asignar el contenido mixto (texto y nodos secundarios) automáticamente en el mismo orden que el definido en el archivo XML de *origen*. Estas conexiones aparecen como líneas de puntos al nivel del nodo principal (*véase el elemento <para> de la asignación*). En este apartado explicamos cómo asignar contenido mixto. También verá el efecto que tiene usar conexiones estándar (basadas en destino) con contenido mixto.

Nota: Las conexiones basadas en origen también se pueden usar en campos de BD con contenido mixto (*en las ediciones Professional y Enterprise*).

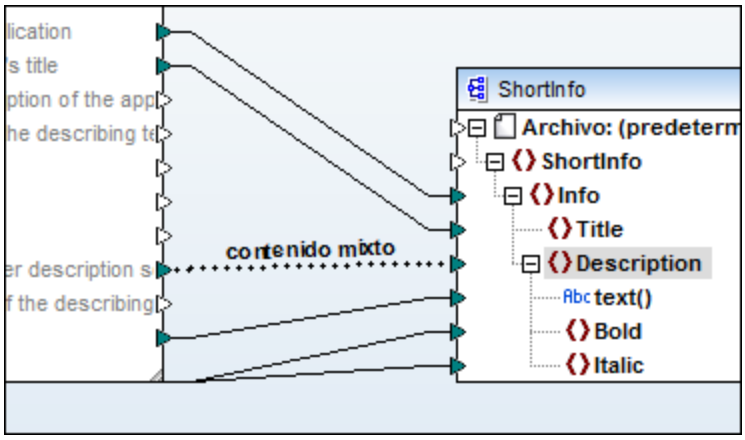
Nota: Para que acepten contenido mixto los componente de destino deben tener nodos de contenido mixto.

Asignar contenido mixto

En este apartado explicamos cómo asignar contenido mixto usando una conexión basada en origen. Para ello necesita estos archivos: **`Tut-OrgChart.xml`**, **`Tut-Orgchart.mfd`**, **`Tut-Person.xsd`** y **`Tut-OrgChart.xsd`**, que encontrará en la carpeta Tutoriales 16.

Instancia XML de origen

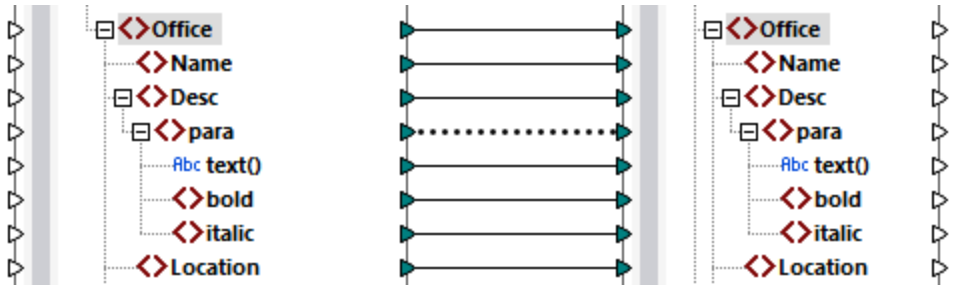
A continuación puede ver un extracto de **`Tut-OrgChart.xml`**. En este ejemplo nos centraremos en el elemento de contenido mixto `<para>` y en sus nodos secundarios `<bold>` e `<italic>`. El elemento **`<para>`** también contiene una instrucción de procesamiento (`<?sort alpha-ascending?>`) y un comentario (`<!--Company details... -->`), que se pueden asignar también, como verá más abajo. Fíjese en la secuencia de los nodos `text` y `bold/italic` en el archivo de instancia XML.



Asignar el elemento <para>

En la imagen siguiente se puede ver una parte de **`Tut-Orgchart.mfd`**. En el ejemplo siguiente las líneas de puntos indican que el elemento `<para>` tiene contenido mixto. Para crear una conexión de contenido mixto siga estos pasos:

1. Seleccione el comando de menú **Conexión | Conectar los secundarios equivalentes**, que conectar lo secundarios equivalentes 53 automáticamente. También puede asignar manualmente el nodo `<para>` a sus nodos secundarios.
2. Conecte el elemento `< para>` del componente de origen con el elemento `< para>` del componente de destino. Aparece un cuadro de diálogo que le pide que defina las conexiones como basadas en el origen.
3. Haga clic en **Sí** para crear una conexión de contenido mixto.
4. Haga clic en el panel **Resultados** para ver el resultado de la asignación. Haga clic en el botón  (**Ajuste automático de línea**) de la barra de herramientas del panel **Resultados** para ver todo el código. Ahora el contenido mixto del nodo `<para>` está asignado en el mismo orden en que aparece en el archivo XML de origen.



Instrucciones de procesamiento y comentarios

Si una asignación tiene instrucciones de procesamiento y/o comentarios y los quiere asignar también, siga estos pasos:

1. Haga clic con el botón derecho en la conexión de contenido mixto (línea de puntos) y seleccione **Propiedades**.
2. En el punto **Basada en origen (contenido mixto)** marque las casillas **Asignar instrucciones de procesamiento** y **Asignar comentarios**.

Conexiones basadas en destino con contenido mixto

Usar conexiones basadas en destino para contenido mixto puede tener consecuencias no deseadas. Para ver cómo afectan estas conexiones al orden de los nodos de contenido mixto siga estas instrucciones:

1. Abra la asignación **`Tut-OrgChart.mfd`** de la carpeta **`Tutoriales`**.
2. Pulse el botón  de la barra de herramientas (Conectar automáticamente los secundarios equivalentes 53). Desmarque la casilla **Crear conexiones de copia total** en la configuración de las conexiones de secundarios equivalentes 53. Esto evita que MapForce cree conexiones de copia total 55 automáticamente.
3. Cree una conexión entre el nodo `para` de origen y el nodo `para` de destino. Aparece un de diálogo que le pide que defina las conexiones como basadas en el origen. Haga clic en **No** para crear una conexión basada en destino.
4. Haga clic en el panel **Resultados** para ver el resultado de la asignación (*imagen siguiente*).

```
 6          <Desc>
 7              <para>The company was established in  in 1995. Nanonull develops nanoelectronic
    technologies for  February 1999 saw the unveiling of the first prototype  The company hopes
    to expand its operations  to drive down operational costs.
 8                  <bold>Vereno</bold><bold>Nano-grid.</bold><italic>multi-core processors.</
    italic><italic>offshore</italic></para>
 9              <para>White papers and further information will be made available in the near
    future.
10              </para>
11          </Desc>
```

En la imagen anterior puede ver que el contenido del elemento `text()` de origen se ha asignado al destino. Sin embargo, el orden de los nodos secundarios(`bold` y `italic`) del resultado sigue el orden que tienen esos nodos en el esquema XML de destino. Esto quiere decir que los elementos `bold` e `italic` no están integrados en el texto sino que se asignan por separado.

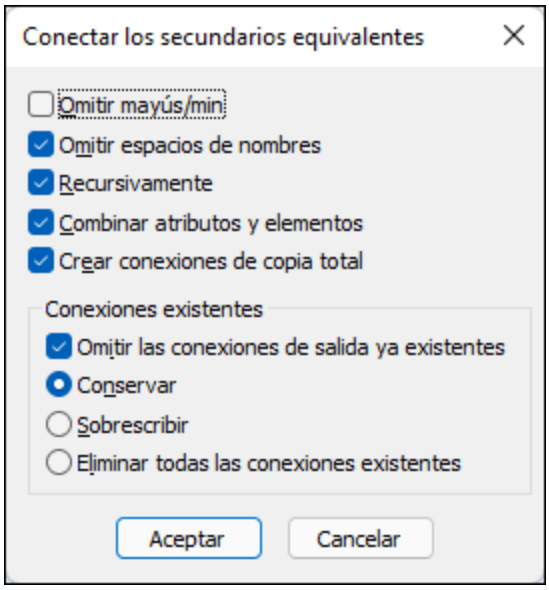
2.2.1.2 Conexiones de secundarios equivalentes

Las conexiones de secundarios equivalentes conectan automáticamente todos los secundarios equivalentes que tengan el mismo nombre en los archivos de origen y de destino. Hay varias maneras de habilitar esta opción:

- Haga clic en el botón  de la barra de herramientas (**Conectar los secundarios equivalentes automáticamente**).
- Vaya al menú **Conexión** y haga clic en **Conectar los secundarios equivalentes automáticamente**.

Configurar las conexiones de secundarios equivalentes

Para cambiar la configuración de los secundarios equivalentes, haga clic con el botón derecho del ratón en cualquier conexión y seleccione la opción **Conectar los secundarios equivalentes** en el menú contextual. También puede ir al menú **Conexión** y hacer clic en **Configurar la conexión de secundarios equivalentes**. Así, se abrirá el cuadro de diálogo **Configurar la conexión de secundarios equivalentes** (*ver imagen siguiente*).



En la lista siguiente se describen las opciones disponibles en el cuadro de diálogo **Configurar la conexión de secundarios equivalentes**. Estas opciones solamente se aplican si está pulsado el botón  de la barra de herramientas (**Alternar la conexión automática de secundarios**).

Opciones de equivalencia

La sección *Opciones de equivalencia* permite relajar criterios de coincidencia y definir cómo compara nombres de nodos. Estas son las opciones disponibles:

- *Usar nombres de columnas para componentes de Excel:* esta opción sólo se aplica a componentes de Excel (*Enterprise Edition*). Esta opción significa que para la comparación se utilizarán los nombres de columna definidos por el usuario (p.ej., `Company`) en lugar de los nombres de referencia de columna

(p.ej., A, B, C). Los nombres de columna definidos por el usuario se establecen en el cuadro de diálogo **Seleccionar rango de celdas** y aparecen como anotaciones en un componente de Excel.
- *Omitir espacios de nombres:* los secundarios equivalentes se conectan independientemente de los espacios de nombres de los nodos secundarios.
- *Combinar atributos y elementos:* esta opción permite crear conexiones entre atributos y elementos que tienen el mismo nombre. Por ejemplo, si existen dos nodos `ID` se crea una conexión aunque uno sea un elemento y el otro un atributo.
- *Omitir mayús/min:* los secundarios equivalentes se conectan independientemente de si sus nombres son iguales pero no coinciden en el uso de mayúsculas y minúsculas.
- *Combinar sólo caracteres alfanuméricos:* si esta opción está activa, se compararán sólo dígitos y letras. Otros caracteres, como espacios, comas, puntos, etc. se descartarán antes de la comparación.

Descendentes
La sección *Descendentes* describe cómo proceder con nodos descendentes/secundarios. Estas son las opciones disponibles:

- *Crear conexiones de copia total:* esta casilla está activada por defecto. Es una opción que crea (si es posible) conexiones de copia total 55 que crean asignaciones de datos entre nodos que tienen nodos secundarios que son muy parecidos o idénticos. Una conexión de copia total, representada por una línea gruesa, evita el desorden y facilita la comprensión de la asignación.
- *Recursivamente:* esta opción crea conexiones nuevas entre secundarios equivalentes si estos tienen el mismo nombre. En este caso no se tiene en cuenta la profundidad de anidación de los nodos.

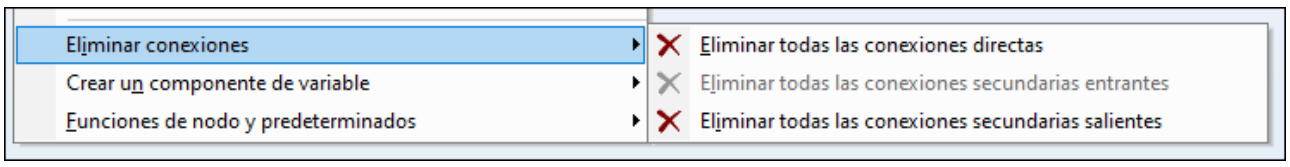
Conexiones existentes
La sección *Conexiones existentes* especifica qué se puede hacer con conexiones existentes. Estas son las opciones disponibles:

- *Omitir las conexiones de salida ya existentes:* esta opción crea más conexiones para todos los secundarios equivalentes, aunque ya tengan conexiones de salida.
- *Conservar:* esta opción mantiene las conexiones que ya existen.
- *Sobrescribir:* esta opción sobrescribe las conexiones que ya existen.
- *Eliminar todas las conexiones existentes:* esta opción elimina todas las conexiones existentes antes de crear conexiones nuevas.

Eliminar conexiones en grupo

Si quiere eliminar conexiones en grupo siga estos pasos:

1. Haga clic con el botón derecho en un nombre de nodo dentro del componente.
2. Seleccione **Eliminar conexiones | Eliminar todas las conexiones <...>** en el menú contextual (*imagen siguiente*).



- *Eliminar todas las conexiones directas:* esta opción elimina todas las conexiones que están asignadas directamente a o desde el nodo seleccionado.

- *Eliminar todas las conexiones secundarias entrantes:* esta opción se activa solamente si hace clic con el botón derecho del ratón en un nodo principal del componente de destino. Esta opción elimina todas las conexiones secundarias entrantes del nodo principal seleccionado.
- *Eliminar todas las conexiones secundarias salientes:* esta opción se activa solamente si hace clic con el botón derecho del ratón en un nodo principal del componente de origen. Elimina todas las conexiones secundarias salientes del nodo principal seleccionado.

2.2.1.3 Conexiones de copia total

Las conexiones de copia total crean asignaciones de datos entre nodos que tienen nodos secundarios que son muy parecidos o idénticos. Sólo se pueden establecer este tipo de conexiones si los dos formatos son idénticos (p.ej. JSON a JSON o XML a XML). Este principio también se aplica a todos los componentes de texto: archivos planos, FlexText y archivos EDI. Todos estos formatos son archivos de texto, por lo que puede combinar cualquiera de ellos y crear una conexión de copia total entre los archivos EDI y FlexText, por ejemplo.

La ventaja principal de las conexiones de copia total es que simplifican visualmente el área de trabajo: se crea una conexión, representada por una línea gruesa, en lugar de muchas conexiones (*imagen siguiente*). En los apartados siguientes explicamos cómo crear conexiones de copia total automáticamente y manualmente.

Crear conexiones de copia total automáticamente

Para crear una conexión de copia total automática:

1. Vaya al menú **Conexión**.
2. Haga clic en **Configurar la conexión de secundarios equivalentes**.
3. Marque la casilla **Crear conexiones de copia total** y haga clic en **Aceptar**.
4. Pulse el botón **Alternar la conexión automática de secundarios** de la barra de herramientas. También puede ir al menú **Conexión** y hacer clic en el comando **Conectar automáticamente los secundarios equivalentes**.

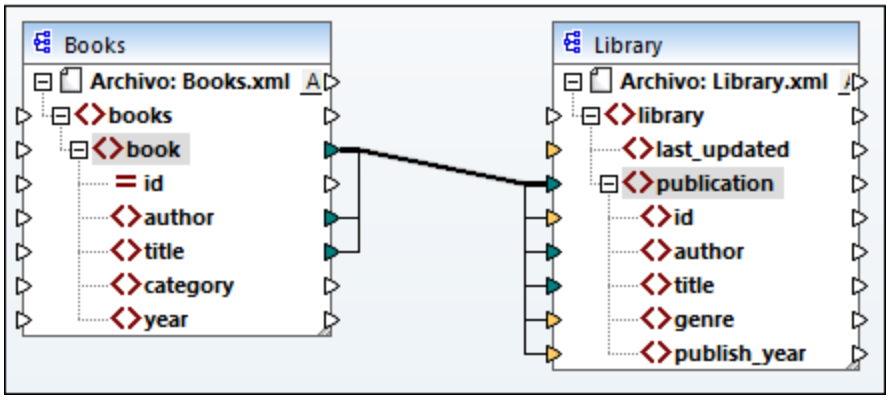
Si el tipo y/o los nombres de los nodos secundarios son distintos, la conexión de copia total no se crea automáticamente, sino que tiene que crearla manualmente.

Crear conexiones de copia total manualmente

Para crear una conexión de copia total manualmente:

1. Agregue un archivo de origen: En el menú **Insertar** haga clic en **Archivo o esquema XML** y navegue hasta **`Books.xml`**,que está en la carpeta de Tutoriales básicos 16.
2. Agregue un archivo de destino: En el menú **Insertar** haga clic en **Archivo o esquema XML** y navegue hasta **`Library.xsd`**, que está en la misma carpeta que **`Books.xml`**. Haga clic en **Omitir** cuando MapForce le pida que indique un archivo XML de muestra.
3. Asigne el nodo `<book>` del componente **`Books`** al nodo `<publication>` del componente **`Library`**. Como las estructuras de `<book>` y `<publication>` no coinciden, no se crea la conexión de copia total. Lo que sí se activa es la función **Conectar automáticamente los secundarios equivalentes**, que conecta automáticamente todos los nodos secundarios que tienen el mismo nombre, como explicamos en el tutorial nº 1 84.

4. Para cambiar la conexión automática a una conexión de copia total haga clic con el botón derecho entre <book> y <publication> y seleccione **Copia total (copia los elementos secundarios)** en el menú contextual.
5. Aparece una ventana de texto que sugiere que reemplace las conexiones existentes con una conexión de copia total. Haga clic en **Aceptar**. Ahora hay una conexión de copia total entre el origen y el destino (*imagen siguiente*).



En la asignación anterior sólo dos nodos secundarios son idénticos en las dos estructuras: <author> y <title>. Esto quiere decir que existe una conexión de copia total entre esos nodos. Los nodos secundarios que no son idénticos no se pueden conectar. En la imagen se puede ver que el elemento id no se incluye en la conexión de copia total porque no es del mismo tipo en el origen y el destino: id es un atributo en el origen y un elemento en el destino. Si intenta crear una conexión entre nodos que no son iguales, como <category> y <genre>, MapForce le pide que reemplace o duplique la entrada.

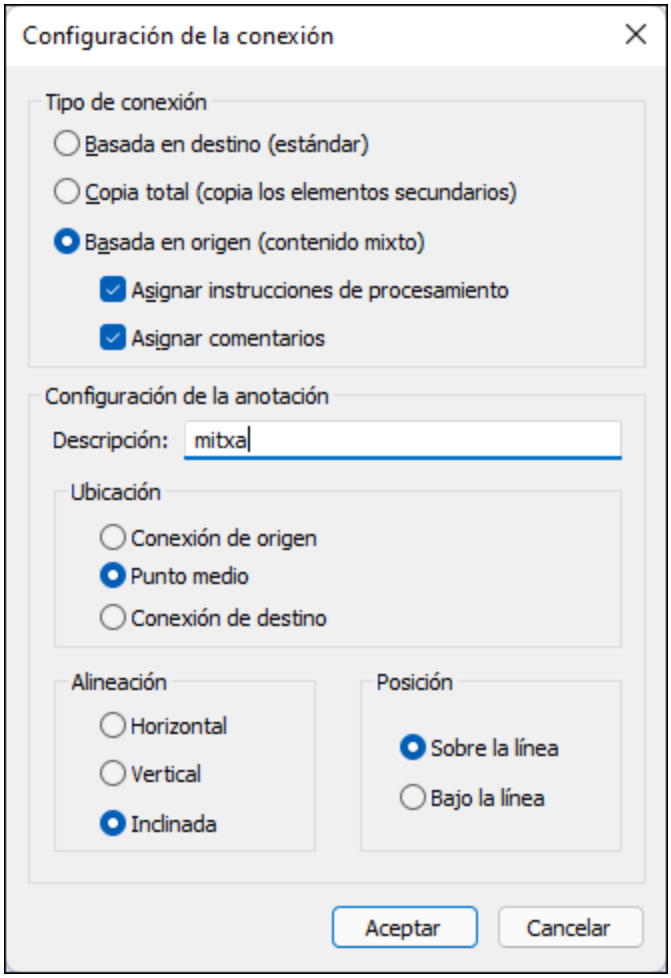
Duplicar la entrada 40 sólo es útil si quiere que el destino acepte datos de más de una entrada, algo que no es necesario en este caso. Si elige reemplazar la conexión de copia total aparece un mensaje que le pide que resuelva o elimine esta conexión. Haga clic en **Resolver la conexión de copia total** si quiere reemplazarla con conexiones basadas en el destino 50 . Si prefiere eliminar la conexión de copia total por completo haga clic en **Eliminar conexiones secundarias**.

Importante

Cuando se crea una conexión de copia total entre un esquema y un parámetro de una función definida por el usuario 205, los dos componentes deben basarse en el mismo esquema. Sin embargo, no es necesario que los dos tengan el mismo elemento raíz.

2.2.2 Configuración de la conexión

Este cuadro de diálogo sirve para configurar conexiones. Para abrirlo haga doble clic en una conexión. También puede hacer clic con el botón derecho en la conexión y seleccionar **Propiedades** en el menú contextual. Los ajustes se dividen en dos partes: tipos de conexión y configuración de la anotación. Para más información siga leyendo.



Tipos de conexión

Puede elegir uno de los tipos de conexión que describimos a continuación:

- Basada en destino (estándar)[50]: estas conexiones se pueden usar en casi todas las asignaciones.
- Copia total (copia lo elementos secundarios)[55]: si un componente de origen y uno de destino tienen los mismos nodos o nodos parecidos con secundarios equivalentes se crea una conexión de copia total automáticamente entre los nodos equivalentes.
- Basada en origen (contenido mixto)[50]: estas conexiones asignan el contenido mixto (texto y nodos secundarios) en el mismo orden que el definido en el archivo XML de origen. Si selecciona **Asignar instrucciones de procesamiento** y/o **Asignar comentarios** podrá incluir estos grupos de datos en el archivo de salida (*imagen siguiente*).

Configuración de la anotación

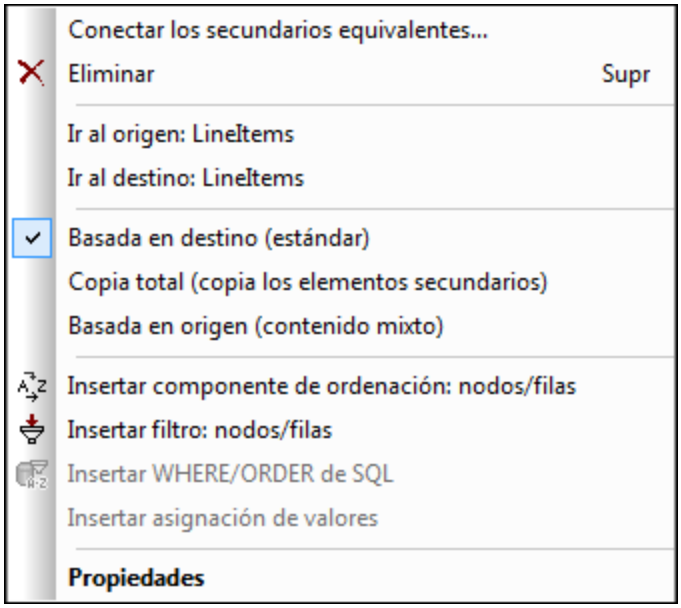
En esta sección puede aplicar etiquetas a las conexiones. Puede aplicar esta opción a todos los tipos de conexión. Para anotar una conexión siga estas instrucciones:

1. Haga clic con el botón derecho en la conexión y seleccione **Propiedades** del menú contextual. O, si lo prefiere, haga doble clic en la conexión.
2. Introduzca el nombre de la conexión seleccionada en el campo **Descripción**. Esto habilita todas las conexiones de la sección **Configuración de la anotación**.
3. Use el resto de grupos para definir las opciones **Ubicación**, **Alineación** y **Posición** de la etiqueta.
4. Pulse el botón  de la barra de herramientas (**Ver anotaciones**). Si no ve este botón en la barra de herramientas puede activarlo en la barra de herramientas **Opciones de vista**.

Nota: si el botón **Ver anotaciones de la barra de herramientas** no está activo sigue pudiendo ver la anotación si pasa el cursor del ratón por encima de la conexión. Si el botón  (Ver información rápida) de la barra de herramientas **Opciones de vista** está habilitado, la anotación aparece como información rápida.

2.2.3 Menú contextual de las conexiones

En este apartado explicamos los comandos que tiene disponibles en el menú contextual de las conexiones. Cuando se hace clic con el botón derecho en una conexión aparece este menú contextual:



Para más información siga leyendo.
Configuración general

- *Conectar los secundarios equivalentes:* Abre el cuadro de diálogo Conectar los secundarios equivalentes[53]. Este comando se habilita si la conexión puede tener secundarios equivalentes.
- *Eliminar:* Elimina la conexión seleccionada.
- *Ir al origen: <nombre del elemento>:* Resalta el conector de salida[31] de la conexión seleccionada.
- *Ir al destino: <nombre del elemento>:* Resalta el conector de entrada[31] de la conexión seleccionada.

Tipos de conexión
Consulte los detalles sobre los tipos de conexión en Tipos de conexión[50] y Configuración de las conexiones[56].

Comandos de inserción

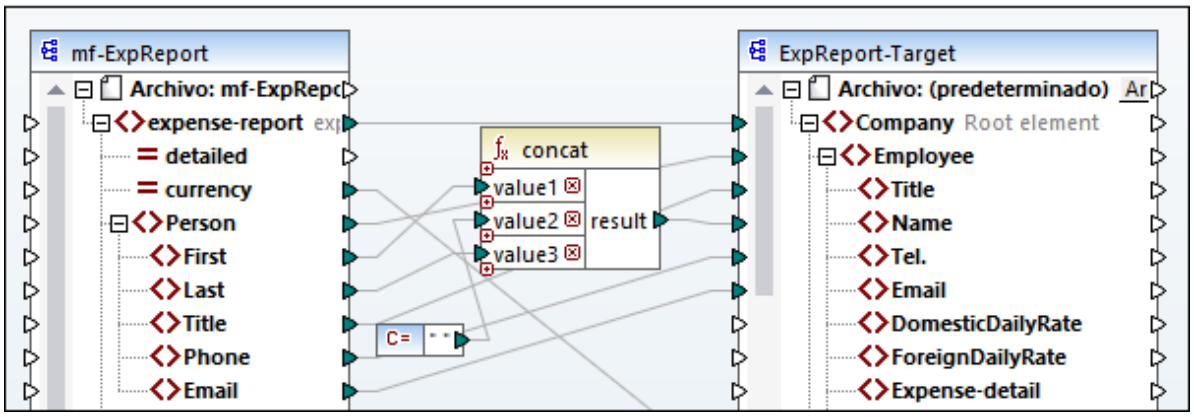
- *Insertar componente de ordenación: nodos/filas:* Agrega un componente de ordenación[172] entre un nodo de origen y uno de destino.
- *Insertar filtro: nodos/filas:* Agrega un componente de filtrado[178] entre un nodo de origen y uno de destino.
- *Insertar componente WHERE/ORDER de SQL/NoSQL:* Agrega un componente WHERE/ORDER de SQL/NoSQL entre un nodo de origen y uno de destino (*en las ediciones Professional y Enterprise*).
- *Insertar asignación de valores:* Agrega una asignación de valores[184] entre un nodo de origen y uno de destino.

Propiedades
Abre el cuadro de diálogo Configuración de las conexiones[56].

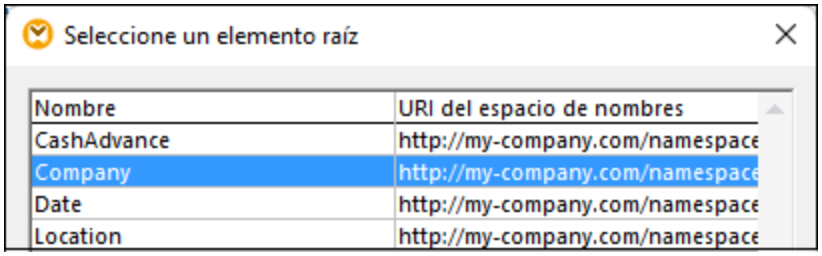
2.2.4 Conexiones defectuosas

En algunas situaciones puede que quiera cambiar el esquema de un componente de origen o de destino. Cambiar esquemas puede afectar a la validez de la asignación y resultar en conexiones defectuosas. En este apartado explicamos cómo reparar esas conexiones después de cambiar el archivo del esquema. Siga las instrucciones del ejemplo siguiente para entender mejor cómo proceder si encuentra conexiones defectuosas.

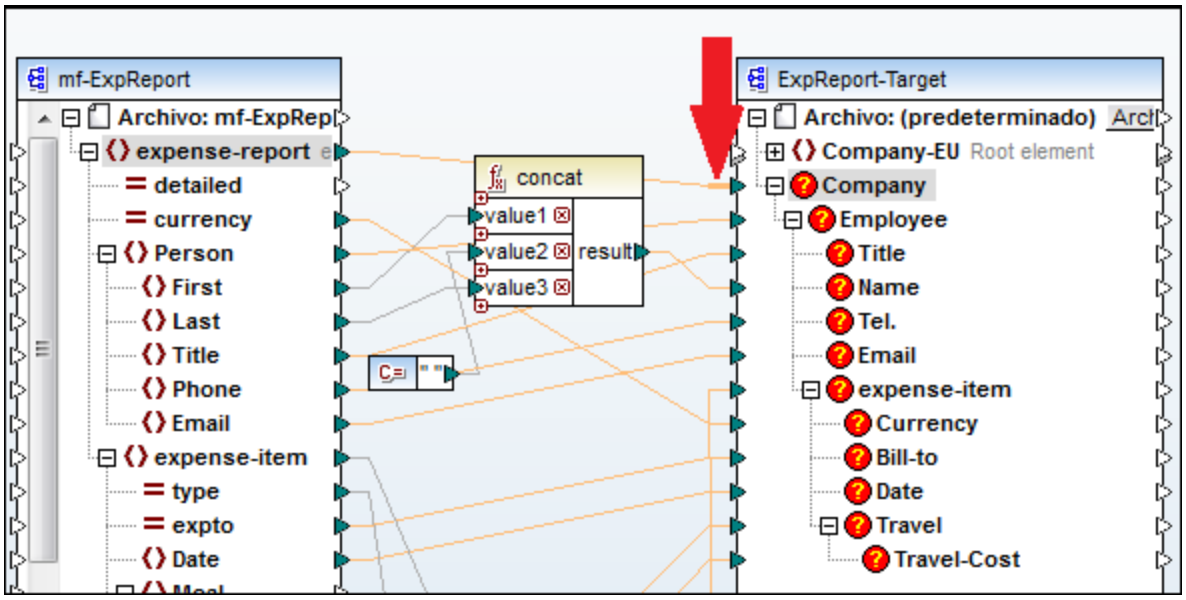
1. Abra la asignación **`Tut-ExpReport.mfd`**, que encontrará en la carpeta Tutoriales [16]. A continuación puede ver parte de esta.



2. Abra **`ExpReport-Target.xsd`** en un editor (p.ej. Altova XMLSpy) y cambie el elemento raíz `Company` en el esquema de destino a `Company-EU`. No es necesario que cierre MapForce.
3. Una vez haya editado el elemento raíz del esquema de destino MapForce le informará de que se han cambiado archivos. Haga clic en el botón **Volver a cargar**. Al haber cambiado el elemento raíz, en el componente hay varias conexiones defectuosas.
4. Haga clic en **Seleccionar un elemento raíz nuevo** en la parte superior del componente (*imagen siguiente*). También puede cambiar el elemento raíz haciendo clic con el botón derecho en el encabezado del componente y seleccionando **Cambiar elemento raíz** en el menú contextual.



5. Seleccione `Company-EU` como nuevo elemento raíz y haga clic en **Aceptar**. El elemento raíz `Company-EU` ahora se puede ver en la parte superior del componente.
6. Ahora tiene que mover la conexión del nodo `Company` al nuevo elemento raíz. Pulse y mantenga pulsada la parte gruesa de la conexión `Company` (*véase la flecha roja más abajo*). Después arrastre la conexión hasta el elemento raíz `Company-EU`.



Un cuadro de diálogo le preguntará si quiere mover todos los nodos secundarios equivalentes. Puede elegir entre mover solamente la conexión seleccionada o mover la conexión seleccionada y aquellos de sus nodos secundarios que tengan equivalentes en el elemento raíz nuevo. En nuestro ejemplo hemos elegido la opción **Incluir conexiones de descendientes**. En cuanto haga clic en este botón los nodos defectuosos desaparecerán del componente.

Nota: Si el nodo al que está asignando los datos tiene el mismo nombre que el nodo de origen pero un espacio de nombres distinto, el cuadro de diálogo de la notificación incluirá el botón **Incluir descendientes y asignar espacio de nombres**. Si hace clic en este botón se mueven las conexiones secundarias del mismo espacio de nombres que el nodo primario de origen a los mismos nodos secundarios del espacio de nombres distinto.

Una solución alternativa

Una solución alternativa al problema que presentamos sería eliminar los nodos defectuosos que ya no necesita en la asignación. Por ejemplo, al eliminar la conexión entre la función **`concat`** y `Name`, el nodo `Name` desaparece del componente `ExpReport-Target`.

Conexiones defectuosas en bases de datos (en las ediciones Professional y Enterprise)

Si un componente de BD tiene conexiones defectuosas deberá cambiar la configuración de los componentes 39. Haga clic en el botón **Cambiar** del cuadro de diálogo **Configuración del componente** para seleccionar una BD distinta o cambiar las tablas del componente de BD. Todas las conexiones y datos de BD válidos/correctos se mantienen si selecciona una BD que tenga la misma estructura.

2.2.5 Conservar conexiones tras eliminación de componentes

MapForce permite conservar conexiones (secundarias) aunque se hayan eliminado varios componentes de transformación 32, como variables, filtros o componentes de ordenación, asignaciones de valores, componentes de entrada simples o componentes SQL/NoSQL-WHERE/ORDER. Las conexiones pueden ser

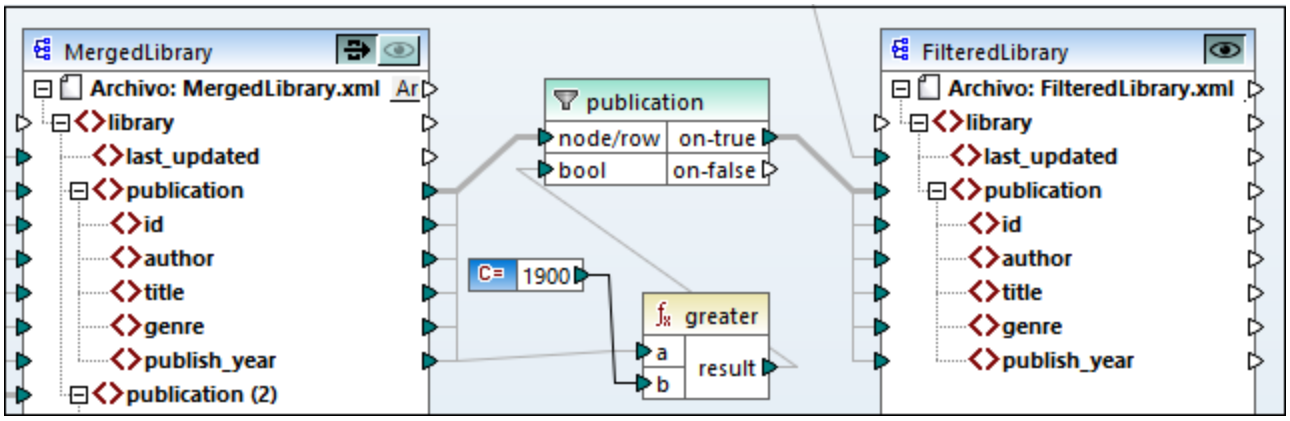
individuales o múltiples. Mantener conexiones puede ser especialmente útil si tiene varias conexiones secundarias, ya que no tiene que restaurar cada una de ellas manualmente después de eliminar el componente de transformación en cuestión. Para habilitar esta característica vaya a **Herramientas | Opciones | Edición** y seleccione **Eliminación inteligente de componentes** (conservar conexiones útiles). Esta opción está deshabilitada por defecto, es decir, si no la activa y elimina un componente de transformación también eliminará todas sus conexiones directas.

Ejemplo

Para ilustrar la eliminación inteligente de componentes hemos usado el archivo de muestra **`Tut3-ChainedMapping`**. Puede encontrarlo en la carpeta de tutoriales básicos [16]:

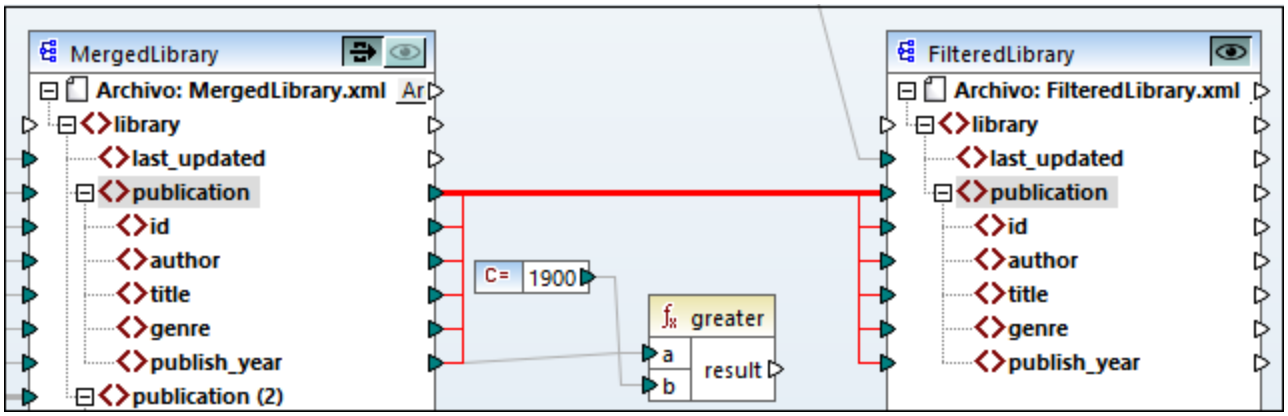
Antes de eliminar el componente

En la imagen siguiente se ve que existen conexiones de copia total [55] entre el componente `MergedLibrary` y el filtro `publication`, así como entre el filtro `publication` y el componente `FilteredLibrary`. Lo que queremos es eliminar el filtro `publication` pero mantener las conexiones de copia total. Para ello marque la casilla **Eliminación inteligente de componentes** del cuadro de diálogo **Opciones** (*imagen anterior*).



Después de eliminar el componente

Una vez haya eliminado la función `publication` se crea la conexión de copia total entre el nodo `publication` de `MergedLibrary` y el nodo `publication` de `FilteredLibrary` (*imagen siguiente*).



Nota: si un componente de filtrado tiene conectadas las dos salidas (`on-true` y `on-false`), entonces se conservan las conexiones secundarias de ambas salidas.

2.3 Procedimientos y funciones generales

Además de crear asignaciones, también puede validar asignaciones y sus resultados, generar código, usar la vista texto y configurar asignaciones. En esta sección encontrará:

2.3.1 Validación

En este apartado explicamos cómo validar asignaciones. También como acceder a una vista previa del resultado, guardarlo y validarlo.

Validar asignaciones de datos

MapForce valida las asignaciones automáticamente cuando se hace clic en el panel **Resultados**. También puede validarlas manualmente, lo que ayuda a identificar y corregir errores potenciales y advertencias antes de ejecutar la asignación. Para validar una asignación manualmente, haga clic en el panel **Asignación** y siga estos pasos:

- En el menú **Archivo** haga clic en **Validar asignación**.
- En la barra de herramientas haga clic en  (**Validar**).

Para validar la asignación MapForce comprueba, entre otras muchas cosas, si faltan conexiones o si hay conexiones incorrectas, si hay tipos de componente incompatibles, etc. Para ver la referencia de todos los estados de validación consulte el apartado de la ventana Mensajes 25. La ventana Mensajes también permite ejecutar acciones relacionadas con los mensajes 25. Para ver los resultados de cada validación por separado, basta con hacer clic en las diferentes pestañas numeradas situadas en el lateral izquierdo de la ventana **Mensajes**. Esta característica es muy práctica cuando se trabaja con varios archivos de asignación simultáneamente.

Validación de los componentes de transformación
A continuación explicamos cómo se validan los componentes de transformación 33:

- Si algún **conector de entrada** obligatorio está desconectado, se genera un mensaje de error y se interrumpe la transformación.
- Si algún **conector de salida** está desconectado, se genera una advertencia y continúa el proceso de transformación. El componente infractor y sus datos se pasan por alto y los datos no se asignan al de salida.

Vista previa y validación de resultados

En MapForce puede incluso consultar una vista previa de los resultados sin necesidad de ejecutar ni compilar el código generado con un procesador o compilador externo. Por lo general, se recomienda consultar la vista previa de resultados de la transformación en MapForce antes de procesar el código generado en una aplicación

externa. Para generar la vista previa de los resultados MapForce ejecuta la asignación y rellena el panel Resultados [28].

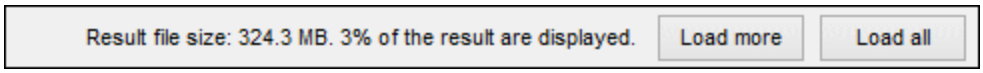
Una vez disponibles en el panel **Resultados**, los datos se pueden validar y guardar. También puede usar el comando **Buscar** (**Ctrl+F**) para encontrar un patrón de texto en el archivo de salida. Para más información consulte Búsqueda en la vista Texto [72]. Los mensajes de error, advertencia e información relacionados con la ejecución de la asignación aparecen en la ventana Mensajes [25].

Si selecciona C++, C# o Java (*ediciones Professional y Enterprise*) como lenguajes de transformación [17], MapForce ejecuta la asignación con su motor de transformación integrado y muestra el resultado en el panel **Resultados**.

Para guardar el resultado de la asignación haga clic en el panel **Resultados** y siga estos pasos:

- Haciendo clic en el comando de menú **Resultados | Guardar el archivo de salida**.
- Haciendo clic en el botón  (**Guardar resultado generado**) de la barra de herramientas.

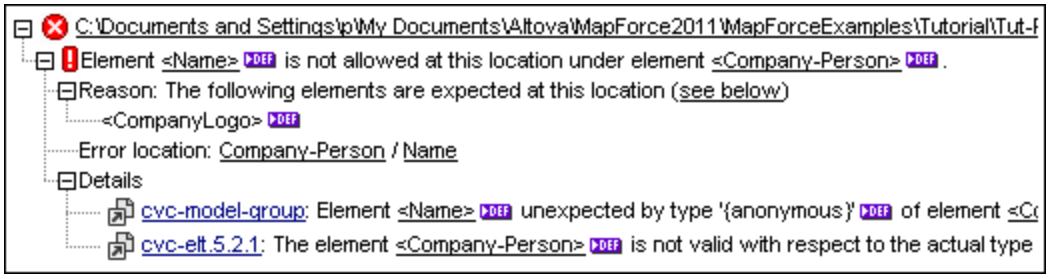
Cargar opciones
A la hora de generar una previa de archivos de gran tamaño, MapForce limita la cantidad de datos que se presentan en el panel **Resultados**. En concreto se presenta solamente parte del archivo y aparece el botón **Cargar más** en la parte inferior del panel (*imagen siguiente*). Al hacer clic en **Cargar más**, se añaden más datos. La vista previa también se puede configurar en la pestaña **Generales** del cuadro de diálogo **Opciones**. Para más información consulte Cambiar opciones de MapForce [472].



Validar resultados de la asignación
Al hacer clic en el panel **Resultados**, aparece una vista previa del resultado de la asignación, que puede validar usando el esquema que tenga asociado. El botón **Validar archivo de salida** y su correspondiente comando de menú (**Resultados | Validar el archivo de salida**) solamente se habilitan si el archivo de salida admite la validación con un esquema. Los resultados de la validación aparecen en la ventana **Mensajes**. Para validar el resultado tiene dos opciones:

- Abrir el panel **Resultados** y hacer clic en el botón  (Validar archivo de salida) de la barra de herramientas.
- Abrir el panel **Resultados** y hacer clic en **Validar el archivo de salida** del menú **Resultados**.

La imagen siguiente ilustra una validación fallida. La ventana **Mensajes** contiene información detallada sobre los errores- Por ejemplo, si hace clic en el enlace `<Name>`, MapForce resaltará este elemento en el panel **Resultados**.



2.3.2 Generación de código

El generador de código es una función integrada en MapForce que le permite generar código a partir de archivos de asignación. Puede utilizar el código generado para ejecutar sus asignaciones de datos fuera de MapFprce, lo que le permitirá automatizar sus operaciones de asignación. Estos son los lenguajes de transformación de datos[17] en los que puede generar código:

- XSLT 1.0/XSLT 2.0/XSLT 3.0 (*todas las ediciones*)
- XQuery (*ediciones Professional y Enterprise*)
- Java (*ediciones Professional y Enterprise*)
- C# (*ediciones Professional y Enterprise*)
- C++ (*ediciones Professional y Enterprise*)

Puede generar código a partir de un solo diseño de asignación (**.mfd**) o a partir de un proyecto de asignación (**.mfp**). Sin embargo, sólo puede generar código a partir de un proyecto en las ediciones Enterprise y Professional. Para más detalles siga leyendo.

Puntos importantes
A la hora de generar código debe tener en cuenta los siguientes aspectos:

- Algunas funciones de MapForce no son compatibles con el código de programa generado. Para más detalles consulte el apartado Funciones compatibles en el código generado[487].
- Para obtener más información sobre la gestión de rutas de acceso en el código generado consulte el apartado Rutas de acceso según el entorno de ejecución[44].
- *Las ediciones Enterprise y Professional:* En la sección *Generación* del cuadro de diálogo **Opciones** puede configurar las opciones generales de generación de código.
- *Las ediciones Enterprise y Professional:* La compatibilidad con las conexiones a bases de datos varía según la plataforma, y hay tipos de conexión que no son compatibles con todas las plataformas. Si su asignación se conecta a una base de datos, elija una conexión de base de datos que sea compatible con el entorno de destino para el que genera el código.

Información de compatibilidad
En la siguiente tabla puede ver un resumen de la información de compatibilidad con los lenguajes de programación C++, C# y Java.

Lenguaje de destino	C++	C#	Java
Entornos de desarrollo	Microsoft Visual Studio 2013, 2015, 2017, 2019, 2022	Microsoft Visual Studio 2013, 2015, 2017, 2019, 2022 Marcos de destino: • .NET Framework • .NET Core 3.1 • NET 5.0 • NET 6.0 • NET 8.0	Java SE JDK 8, 11, 17, 21 (incluido OpenJDK) Eclipse 4.4 o superior Apache Ant
Implementaciones XML DOM	MSXML 6.0 Apache Xerces 3	System.Xml	JAXP
API de la base de datos	ADO	ADO.NET	JDBC

Generar código a partir de una asignación de datos

Para generar código a partir de un diseño de asignación (**.mfd**), siga estos pasos:

1. En el cuadro de diálogo **Opciones** (para C# y C++) o en las opciones de la asignación[76], vaya a la sección *Generación* y seleccione las opciones de generación de código relevantes.
2. Haga clic en **Archivo | Generar código en** y seleccione el lenguaje de transformación correspondiente. Otra opción es seleccionar **Archivo | Generar código en el idioma seleccionado**. En este caso, el código se generará en el idioma que haya seleccionado en la barra de herramientas.
3. Seleccione el directorio de destino para los archivos que se deben generar y haga clic en **Aceptar** para confirmar. MapForce genera el código y muestra el resultado de la operación en la ventana Mensajes[25].

Generar código a partir de un proyecto (ediciones Professional y Enterprise)

Puede generar código C# a partir de proyectos de asignación (**.mfp**) que consistan en varios archivos de diseño de asignación (**.mfd**). Tenga en cuenta que para ello todos los archivos del proyecto deben ser compatibles con C#, es decir, todos sus componentes deben serlo también, como se explica en Funciones compatibles en el código generado[487].

Para generar código a partir de un proyecto de asignación, siga las instrucciones a continuación.

1. Abra el proyecto de asignación correspondiente a partir del cual quiere generar el código.
2. Haciendo clic con el botón derecho en el nombre del proyecto en la ventana Proyecto y entonces eligiendo el comando de menú **Propiedades** del menú contextual. Otra opción es hacer clic en el nombre del proyecto y seleccionar el comando **Propiedades** del menú **Proyecto**.
3. Revise y modifique la configuración del proyecto según proceda. Sobre todo debe comprobar que el lenguaje de destino y el directorio de salida están configurados correctamente. Después haga clic en **Aceptar**.
4. En el menú **Proyecto** haga clic en **Generar código para todo el proyecto**.

Independientemente del idioma seleccionado en el cuadro de diálogo **Propiedades del proyecto**, siempre puede optar por generar el código del proyecto en un idioma diferente, seleccionando el comando de menú **Proyecto | Generar código en | <idioma>**.

Los progresos y resultados de la generación de código aparecen en la ventana Mensajes. Por defecto, el nombre de la aplicación que se ha generado es igual al nombre de proyecto. Si el nombre del proyecto contiene espacios, éstos se convertirán en guiones bajos en el código generado. El código se genera por defecto en el directorio donde está ubicado el proyecto de MapForce, en el subdirectorio **output**.

Puede modificar el directorio de salida y/o el nombre del proyecto en el cuadro de diálogo **Propiedades del proyecto**. Si su proyecto de MapForce tiene carpetas, podrá cambiar las opciones de generación de código para cada una de las carpetas por separado: Haga clic con el botón derecho en una carpeta de interés y seleccione **Propiedades** del menú contextual. Sino todas las carpetas del proyecto heredan por defecto las opciones de generación de código del nivel superior.

Siguientes pasos

Los siguientes pasos varían en función del lenguaje de transformación que haya seleccionado para la generación de código. Si ha generado código en XSLT 1-3 o XQuery, el siguiente paso consistirá en ejecutar la transformación desde la línea de comandos (*ver los detalles más abajo*).

Si ha generado código Java, C# o C++, los siguientes pasos consistirán en compilar y ejecutar el código generado. También puede modificar el código Java, C# y C++ generado e integrarlo en su código personalizado.

Código XSLT y XQuery
Una vez haya generado el código XSLT 1-3 la carpeta de destino contendrá los siguientes archivos:

1. Un archivo de transformación XSLT que tiene este formato: **<Mapping>MapTo<TargetFileName>.xslt**. <Mapping> es el valor del campo *ApplicationName* en la configuración de la asignación[76]. <TargetFileName> es el nombre del componente de destino de la asignación. Para cambiar este valor abra la configuración del componente y edite el valor del campo *Nombre del componente*. Para más información consulte Cambiar configuración de los componentes[39] y Rutas de acceso según el entorno de ejecución[44].
2. Un archivo **DoTransform.bat**, que permite ejecutar la transformación XSLT con Altova RaptorXML Server desde la línea de comandos. Para ejecutar este comando necesita tener instalado RaptorXML.

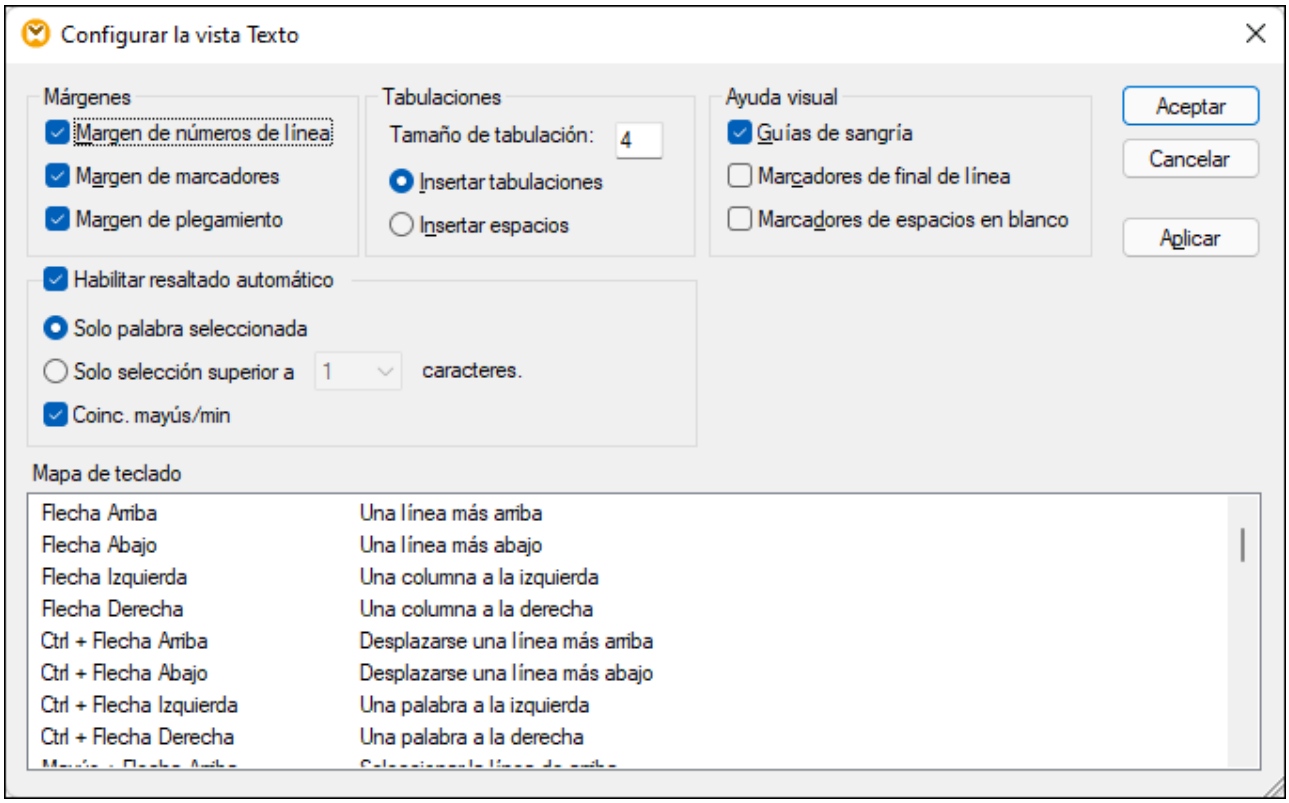
Si se trata de una asignación encadenada[94], se generará un archivo de transformación diferente para cada componente de destino.

La generación de código XQuery es similar a la generación de código XSLT, con la diferencia de que los archivos de transformación tienen una extensión **.xq** y el siguiente formato: **<Mapping>MapTo<TargetFileName>.xq**.

2.3.3 Características de la vista Texto

Los paneles Resultados[28] y XSLT[26] ofrecen varias funciones visuales para facilitar la consulta y edición del texto, como p.ej. los márgenes, el resaltado de texto, los guías de sangría, los marcadores de final de línea y los marcadores de espacios en blanco. Puede personalizar estas funciones en el cuadro de diálogo Configurar

la vista Texto (*imagen siguiente*). La configuración elegida en este cuadro de diálogo se aplica a toda la aplicación y no sólo al documento activo.



Hay varias maneras de abrir el cuadro de diálogo **Configurar la vista Texto**:

- Con el comando de menú **Resultados | Configurar la vista Texto**.
- Con el botón  (**Configurar la vista Texto**) de la barra de herramientas.
- Haciendo clic con el botón derecho en el panel **Resultados** y seleccionando **Configurar la vista Texto** en el menú contextual.

Algunas características visuales de la **vista Texto** también se pueden activar/desactivar desde la barra de herramientas Vista Texto, desde el menú de la aplicación o con teclas de acceso rápido. Para más información sobre las teclas de acceso rápido consulte el panel **Mapa de teclado** situado en la parte inferior del cuadro de diálogo **Configurar la vista Texto**.

A continuación puede consultar la lista de opciones disponibles.

- Márgenes

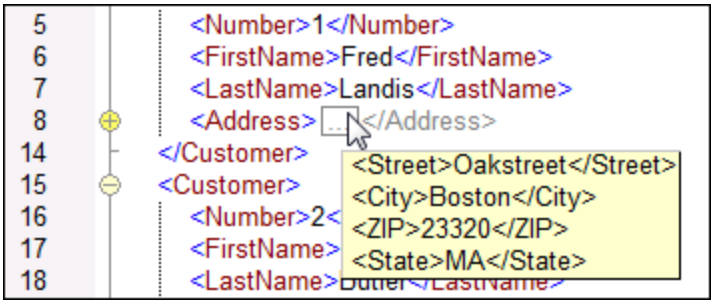
  margen de números de línea
  Los números de línea aparecen en el margen de números de línea (imagen anterior), que se puede activar o desactivar en el cuadro de diálogo **Configurar la vista Texto**. Cuando se contrae una sección de texto, los números de línea del texto contraído también se ocultan.

Margen de marcadores (cuadro de diálogo)
Las líneas del documento se pueden marcar para realizar consultas rápidas más adelante. Si el **margen de marcadores** está activado en en el cuadro de diálogo **Configurar la vista Texto**, los marcadores aparecen en el margen de marcadores (*imagen siguiente*). De lo contrario, las líneas marcadas se resaltan en color cian.

Puede editar marcadores y navegar por ellos con los comandos de la tabla que ve a continuación. Estos comandos también están disponibles en el menú **Resultados** y en el menú contextual que aparece cuando se hace clic con el botón derecho en los paneles **Resultados**, **XSLT** y **XQuery**.

	**Insertar o quitar marcador (Ctrl + F2)**
	**Ir al siguiente marcador (F2)**
	**Ir al marcador anterior (Mayús + F2**
	**Eliminar todos los marcadores (Ctrl + Mayús + F2**

Marcadores de plegamiento de código
La característica plegamiento de código permite expandir y contraer nodos. Los nodos que se pueden expandir/contraer se marcan en el margen de plegamiento. El margen puede activarse o desactivarse en el cuadro de diálogo **Configurar la vista Texto**. Para expandir y contraer porciones de código basta con hacer clic en los iconos + y - situados en el lateral izquierdo del panel. Las porciones de código contraído se señalan con puntos suspensivos (*imagen siguiente*). Para ver el código contraído debe pasar el cursor del ratón sobre los puntos suspensivos. Al hacerlo se abre un cuadro emergente que muestra una vista vista previa, como se ve en la imagen siguiente. No olvide que si la vista previa no cabe en el cuadro emergente, al final del código aparecen puntos suspensivos.



Habilitar el resaltado automático

La opción **Habilitar el resaltado automático** permite visualizar todas las coincidencias del texto seleccionado. La selección se resalta en azul pálido y las coincidencias se resaltan en naranja pálido. La selección y sus coincidencias también se indican con cuadrados grises en la barra de desplazamiento, que también indica la posición actual del cursor por medio de un marcador azul. Puede definir como selección una palabra entera o un número fijo de caracteres. También puede especificar si se deben tener en cuenta o no las mayúsculas y minúsculas.

En el caso de la selección de caracteres, podrá especificar el número mínimo de caracteres que deben coincidir con la selección, empezando por el primer carácter de la selección. Por ejemplo, puede elegir que coincidan dos caracteres o más. Cuando se trate de búsquedas de palabras, la aplicación considera que estos elementos son palabras independientes: nombres de elementos (sin paréntesis angulares), paréntesis angulares de etiquetas de elementos, nombres de atributos y valores de atributos sin comillas.

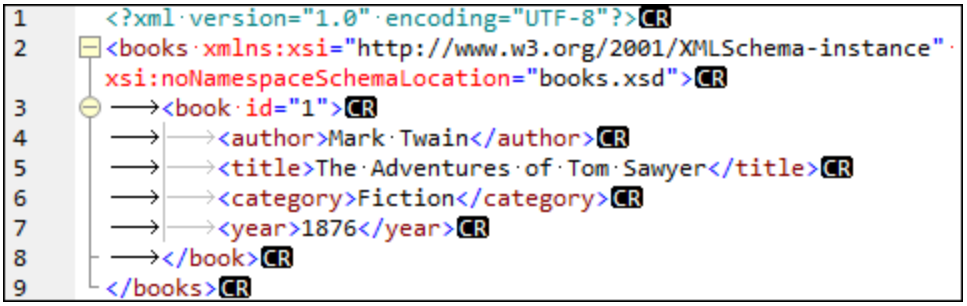
Ayuda visual

Guías de sangría

Las guías de sangría son líneas verticales que indican la longitud de la sangría de una línea. Las guías de sangría se pueden activar o desactivar en el cuadro de diálogo **Configurar la vista Texto**. Las opciones **Insertar tabulaciones** e **Insertar espacios** tienen efecto cuando se ejecuta el comando **Resultados | Texto XML pretty-print**.

Marcadores de final de línea y marcadores de espacios en blanco

Los marcadores de final de línea y los marcadores de espacios en blanco (*imagen siguiente*) se pueden activar o desactivar en el cuadro de diálogo **Configurar la vista Texto**. Cada flecha representa un carácter de tabulación, *CR* representa un retorno de carro y cada punto representa un espacio en blanco.



Más opciones de configuración de la vista Texto

Color de sintaxis
El color de sintaxis es otra ayuda visual que permite leer código más cómodamente. El color de sintaxis se aplica en función del valor semántico del texto. Por ejemplo, en los documentos XML, dependiendo de si el nodo XML es un elemento, un atributo, contenido, una sección CDATA, un comentario o una instrucción de procesamiento, el nombre del nodo (y en ocasiones el contenido del nodo) tendrá un color diferente.

Alejarse y acercarse con el zoom
Puede acercarse y alejarse en la vista Texto si mueve la rueda del ratón mientras pulsa la tecla **Ctrl**. También puede usar las teclas - o + mientras pulsa la tecla **Ctrl**.

Formato pretty-print
El comando **Texto XML pretty-print** ajusta el formato del documento XML que está activo en la **vista Texto** para representarlo de forma estructurada. Por defecto, cada nodo secundario se presenta alejado de su elemento primario por cuatro caracteres de espacio. Las guías de sangría se pueden activar o desactivar en el cuadro de diálogo **Configurar la vista Texto**. Para aplicar formato pretty-print a un documento XML seleccione el comando de menú **Resultados | Texto XML pretty-print** o haga clic en el botón  (**Pretty-print**) de la barra de herramientas.

Ajuste automático de línea
El ajuste automático de línea ayuda a visualizar fragmentos de código dentro del margen del área o la trabajo. Si se deshabilita, hay partes del texto que no son visibles en el área de trabajo. Para activar el ajuste automático de línea en el documento activo use el comando Resultados | Ajuste automático de línea haga clic en el botón  (**Ajuste automático de línea**) de la barra de herramientas.

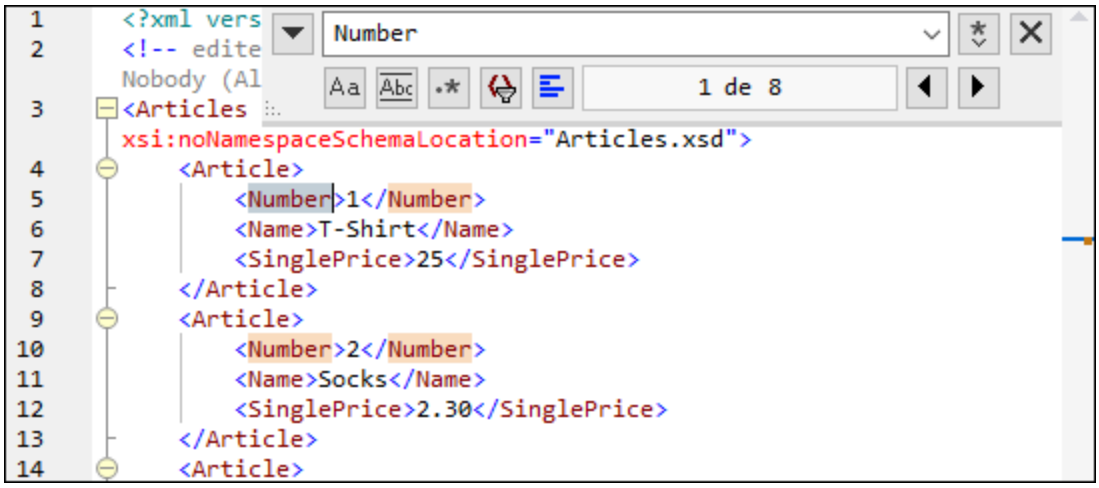
2.3.4 Búsquedas en la vista Texto

Puede realizar búsquedas en el texto de los paneles **Resultados** y **XSLT** con varias opciones y ayudas visuales.

Para iniciar una búsqueda pulse **Ctrl+F** (o seleccione el comando de menú **Edición | Buscar**). La búsqueda puede realizarse en todo el documento o dentro de una selección de texto.

Puede introducir una cadena de texto o seleccionar una de las últimas diez búsquedas en el cuadro combinado. Cuando introduzca o seleccione la cadena de búsqueda, todas las coincidencias se resaltarán y las posiciones de los resultados se indican en la barra de desplazamiento por medio de marcadores color beige. El resultado que esté seleccionado se resalta en gris.

El resultado que esté seleccionado se resalta en un color distinto al de los demás resultados y su posición se indica en la barra de desplazamiento por medio de un marcador de cursor azul oscuro. El número total de resultados aparece debajo del término de búsqueda, además de la posición de índice del resultado que esté seleccionado. Puede recorrer los resultados en ambos sentidos con los botones  **(Anterior, Mayús+F3)** y  **(Siguiente, F3)** situados en la esquina inferior derecha.



Para realizar búsquedas dentro de una selección: (i) marque la selección, (ii) active la opción **Buscar en la selección** para bloquear la selección e (iii) introduzca el término de búsqueda. Para buscar dentro de otra selección, desbloquee la selección actual desactivando la opción **Buscar en la selección**, marque una nueva selección y active otra vez la opción **Buscar en la selección**.

Opciones de búsqueda

Los criterios de búsqueda pueden configurarse con los botones situados debajo del campo del término de búsqueda. Si una opción está activada, su botón aparece en color azul. Estas son las opciones de búsqueda disponibles:

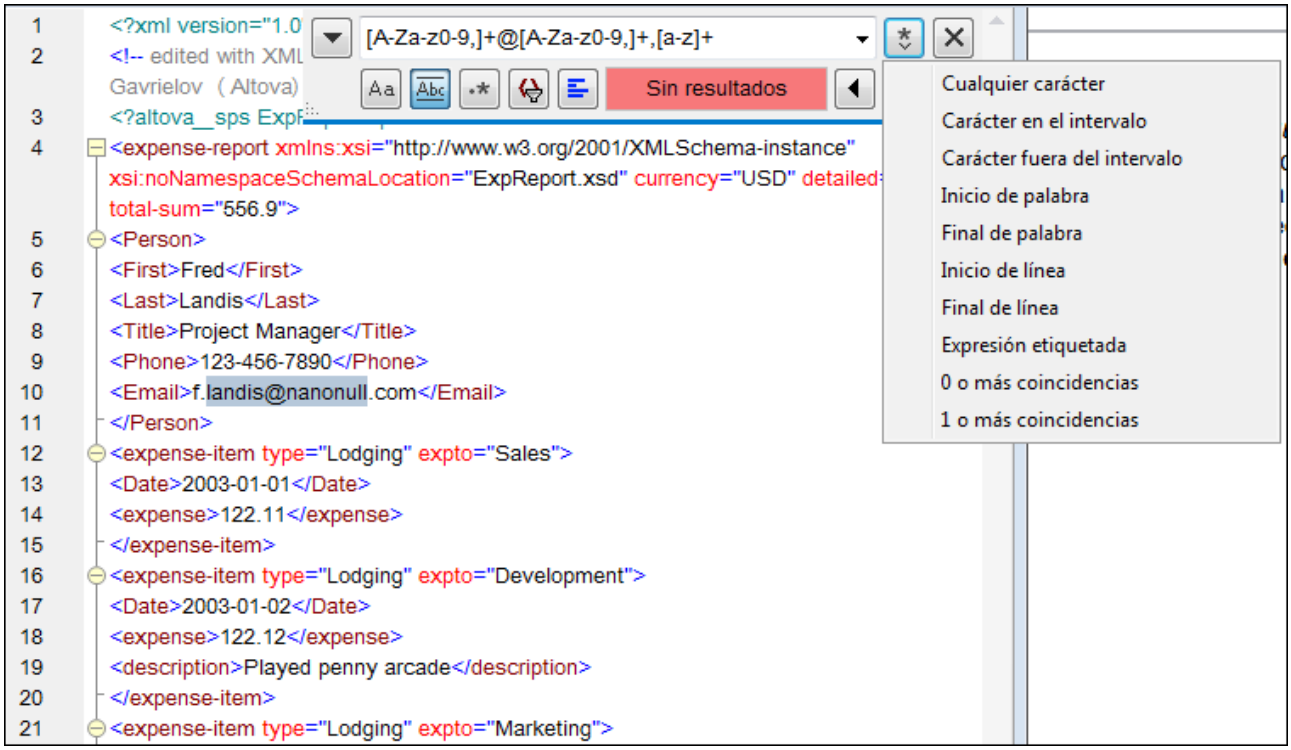
Opción	Icono	Descripción
Coinc. mayús/min	Aa	La búsqueda tiene en cuenta las mayúsculas y minúsculas a la hora de realizar la búsqueda (`Address` no es lo mismo que `address`).
Sólo palabras completas	Abc	Solo se consideran coincidencias las palabras completas.
Expresión regular	.*	Si activa esta opción, el término de búsqueda se lee como expresión regular (*véase más abajo*).
Buscar delimitador		Cuando se introduce un término de búsqueda, los resultados de la búsqueda se resaltan y uno de ellos se marcará como selección actual. Con la opción **Buscar delimitador** puede definir si esta primera selección se hace en relación a la posición actual del cursor o no. Es decir, si la opción **Buscar delimitador** está activada, entonces el primer resultado seleccionado será el siguiente resultado a partir de la posición actual del cursor. Por el contrario, si la opción **Buscar delimitador** está desactivada, el primer resultado seleccionado será el primer resultado del documento, empezando desde el principio.
Buscar en la selección		Si activa esta opción, la selección actual se bloquea y la búsqueda se ejecuta en la selección solamente. De lo contrario, la búsqueda se ejecuta en todo el documento. Antes de realizar una selección nueva deberá

		desbloquear la selección actual desactivando el botón de la opción **Buscar en la selección**.

Uso de expresiones regulares

Puede usar expresiones regulares (regex) para buscar cadenas de texto en el documento. Para ello lo primero es activar la opción **Expresión regular** .*. Al activar esta opción estamos especificando que el texto del campo del término de búsqueda debe evaluarse como expresión regular. El segundo paso consiste en introducir la expresión regular en el campo de búsqueda. Si necesita ayuda para construir su expresión regular, haga clic en el botón **Generador de expresiones regulares** (situado a la derecha del campo de búsqueda). Seleccione un elemento de la lista desplegable para introducir los caracteres correspondientes en el campo de búsqueda.

A continuación puede ver un ejemplo de expresión regular que se utiliza para buscar direcciones de correo electrónico.



Expresiones regulares personalizadas

A continuación puede ver una lista de metacaracteres de expresión regular compatibles con la función de búsqueda y reemplazo.

.	Cualquier carácter. Es un comodín para un solo carácter.
(abc)	Los metacaracteres (y) marcan el inicio y el final de una expresión regular. Las expresiones regulares pueden serle de utilidad a la hora de etiquetar (es decir, recordar)

	una región concreta del resultado de la búsqueda y poder hacerle referencia más adelante (referencia inversa). Puede etiquetar (y hacer referencia inversa a) un máximo de 9 subexpresiones. Por ejemplo, **`(the) \1`** encuentra la cadena `the the`. Esta expresión significa literalmente: buscar la cadena "the" (y recordarla como región etiquetada), seguida de un espacio, seguida de una referencia inversa a la región etiquetada encontrada previamente.
`\n`	Siendo `n` un número del 1 al 9, `n` hace referencia a la correspondiente región etiquetada (*ver fila anterior*).
`\x`	Permite usar caracteres que de lo contrario tendrían un significado propio. Por ejemplo, `\[` se interpretaría como `[` y no como el inicio de un conjunto de caracteres.
`\<`	Inicio de palabra.
`\>`	Final de palabra.
`\`	Inserta un carácter de escape al carácter que aparece después de la barra diagonal inversa. En otras palabras, la expresión `\x` permite usar el carácter `\` de forma literal. Por ejemplo, `\[` se interpretaría como `[` y no como el principio de un conjunto de caracteres.
`[...]`	Encuentra cualquiera de los caracteres del conjunto. Por ejemplo, **`[abc]`**encuentra los caracteres `a`, `b` o `c`. También puede usar intervalos como **`[a-z]`** para buscar cualquier carácter en minúsculas.
`[^...]`	Encuentra cualquier carácter que no esté en este conjunto. Por ejemplo, **`[^A-Za-z]`** encuentra cualquier carácter excepto caracteres alfabéticos en mayúsculas o minúsculas.
`^`	Encuentra el inicio de línea (a no ser que se use dentro de un conjunto de caracteres, ver fila anterior).
`$`	Encuentra el final de línea. Por ejemplo, **`A+$`** encuentra una A o más de una A que estén al final de una línea.
`*`	Encuentra cero o más instancias de la expresión precedente. Por ejemplo, **`Sa*m`** encuentra `Sm` , `Sam` , `Saam` , `Saaam` , etc.
`+`	Encuentra una o más instancias de la expresión precedente. Por ejemplo **`Sa+m`** encuentra `Sam` , `Saam` , `Saaam` , etc

Buscar caracteres especiales

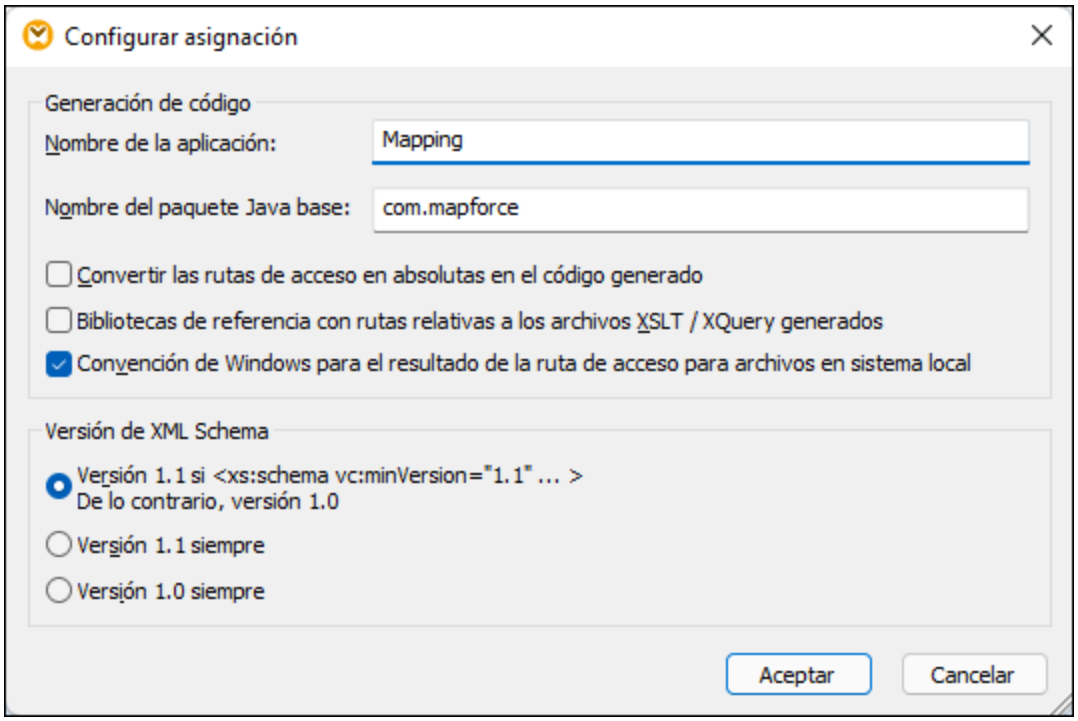
Puede buscar cualquiera de los caracteres especiales siguientes dentro del texto si habilita la opción **Usar expresiones regulares**:

- `\t` (tabulador)
- `\r` (retorno de carro)
- `\n` (línea nueva)
- `\\` (barra inversa invertida)

Por ejemplo, para encontrar un caracter de tabulador pulse **Ctrl + F**, seleccione la opción **Usar expresiones regulares** e introduzca **`\t`** en la caja de diálogo **Buscar**.

2.3.5 Configuración de la asignación

El cuadro de diálogo **Configuración de la asignación** (*imagen siguiente*) permite definir las opciones de los documentos. Para abrirlo, vaya al menú **Archivo** y haga clic en **Configuración de la asignación**. También puede hacer clic con el botón derecho en un área vacía del panel de asignación y seleccionar **Configuración de la asignación** en el menú contextual.



Estas son las opciones que se pueden configurar en este cuadro de diálogo:

- Generación de código
  - *Nombre de la aplicación:* define el prefijo de nombre del archivo XSLT o el nombre de aplicación Java, C# o C++ para los archivos de transformación generados (*sólo disponible en las ediciones MapForce Professional y MapForce Enterprise*).
  - *Nombre del paquete Java base (sólo disponible en las ediciones MapForce Professional y MapForce Enterprise):* esta opción es relevante si escoge Java como lenguajes de transformación. Define el nombre del paquete base para los resultados Java.
  - Convertir las rutas de acceso en absolutas en el código generado: esta casilla afecta a todas las rutas de los componentes de asignación, excepto a las rutas de acceso a los archivos de las bibliotecas externas (como las bibliotecas XSLT). Define si las rutas de acceso de los archivos deben ser relativas o absolutas en el código de programa generado. Consulte Rutas de acceso según el entorno de ejecución 44 para obtener más información.

- *Hacer referencia a bibliotecas con rutas relativas a los archivos XSLT/XQuery generados:* esta casilla se puede usar si el lenguaje de la asignación es XQuery (*sólo disponible en las ediciones MapForce Professional y MapForce Enterprise*) o XSLT. Esta opción es útil si una asignación hace referencia a una biblioteca XSLT o XQuery y quiere generar archivos XSLT o XQuery a partir de esa asignación. Marque esta casilla si quiere que las rutas de la biblioteca sean relativas al directorio del código XSLT o XQuery generado. Si no la marca, las rutas de las bibliotecas serán generadas cuando genere el código. Consulte también Rutas de las bibliotecas en el código generado 44.

- *Convención de Windows para el resultado de la ruta de acceso para archivos en sistema local:* esta casilla se puede usar si el lenguaje de la asignación es XQuery (*sólo disponible en las ediciones MapForce Professional y MapForce Enterprise*), XSLT 2.0 o XSLT 3.0. Marque esta casilla para que MapForce siga la convención de Windows para rutas de acceso. Cuando genere código XSLT 2.0, XSLT 3.0 o XQuery, el nombre del archivo procesado se recupera internamente usando la función **`document-uri`** que devuelve una ruta con el formato `archivo:// URI` para lo archivos locales. Si marca esta casilla, la especificación `archivo:// ruta URI` se convierte automáticamente a una ruta de archivo Windows completa (p.ej. `C:\...`) y así poder seguir con el procesamiento.

Configuración de los archivos de salida (disponible en las ediciones Professional y Enterprise)

En el cuadro combinado **Extremos de línea** puede especificar el extremo de las líneas de los archivos de salida. La opción *Valor predeterminado de la plataforma* se refiere a la opción predeterminada para el sistema operativo de destino (p.ej. CR+LF para Windows, LF para macOS o LF para Linux). También puede seleccionar el extremo de línea que prefiera. Las opciones seleccionadas aquí son de una importancia fundamental para la implementación de archivos de ejecución de MapForce Server (`.mfx`) en servidores FlowForce Server que se ejecuten en sistemas operativos distintos.

Versión de XML Schema

En este grupo de opciones puede indicar qué versión de XML Schema se usa en el archivo de asignación. Sin embargo, tenga en cuenta que no todas las características propias de la versión 1.1 son compatibles con MapForce. Si la declaración `xs:schema vc:minVersion="1.1"` está presente en el esquema, se usa la versión 1.1. De lo contrario se usa la versión 1.0.

Si el documento XSD no tiene el atributo `vc:minVersion` o el valor de `vc:minVersion` no es `1.0` ni `1.1`, entonces el modo predeterminado será XSD 1.0. No se debe confundir el atributo `vc:minVersion` con el atributo `xsd:version`. El primero almacena el número de versión XSD, mientras que el segundo almacena el número de versión del documento. Cuando se cambia esta opción de configuración en una asignación, todos los esquemas que tengan la versión de esquema XML seleccionada se volverán a cargar y puede que la validez de la asignación se vea afectada.

Configuración de la operación de servicio web (edición Enterprise)

Los campos **Definiciones WSDL**, **Servicio**, **Extremo** y **Operación** se rellenan automáticamente si el documento de asignación forma parte de una implementación de servicio web.

3 Tutoriales

Sitio web de Altova: Vídeos de demostración de MapForce

Estos tutoriales le ayudarán a familiarizarse con los procesos de transformación de datos de MapForce. Aprenderá los procesos básicos paso a paso. La complejidad de los tutoriales aumenta gradualmente. Por este motivo recomendamos que los siga en el orden en que los presentamos. Es recomendable tener conocimientos básicos de XML y XML Schema.

Archivos de ejemplo
Los archivos de asignación de datos que aparecen en los tutoriales pueden encontrarse en la carpeta de tutoriales básicos 16. Si no está seguro de cómo podrían afectar ciertos cambios a estos archivos le animamos a que haga copias de seguridad antes de modificarlos.

Tutoriales disponibles

Un archivo de origen a un archivo de destino
En este tutorial 79 aprenderá a usar los mecanismos clave de MapForce para asignar los nodos de un archivo de origen a los de un archivo de destino. El tutorial explica cómo convertir un archivo XML definido por un esquema XML en un archivo XML definido por un esquema XML distinto.
Varios archivos de origen a un solo destino
En este tutorial 89 aprenderá a combinar los datos de distintos archivos XML de origen con los de un único archivo de destino.
Asignaciones encadenadas
En este tutorial 94 crearemos una asignación simple como la del segundo tutorial para después filtrar los datos resultantes y pasarlos a un segundo archivo de destino.
Varios archivo de origen a varios archivos de destino
En este tutorial 103 aprenderá a leer datos de distintos archivos de instancia XML de una misma carpeta y escribir esos datos en distintos archivos XML que se generan sobre la marcha.

3.1 De esquema a esquema

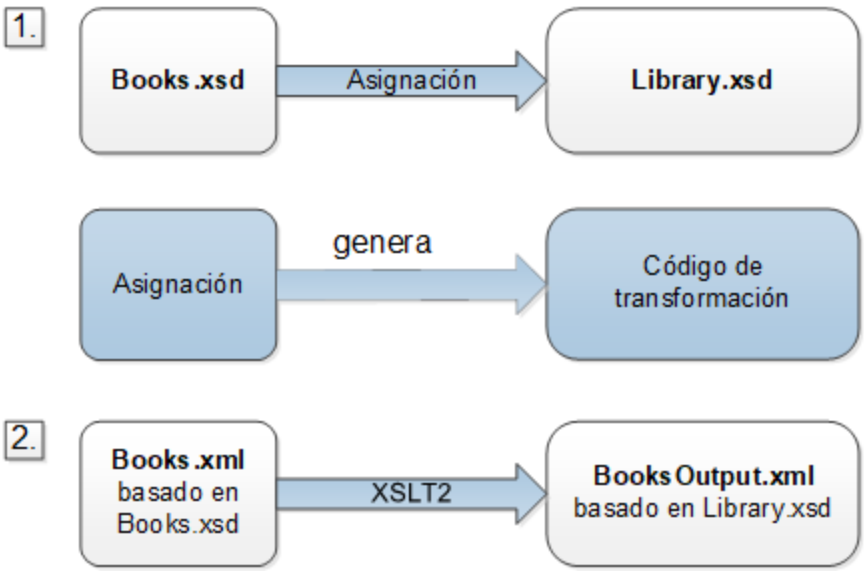
En este tutorial se explica cómo crear una asignación básica. El objetivo es transferir datos de un archivo XML A que tenga asignado un esquema XML A y pasarlos a un archivo XML B que tenga asignado un esquema B. Estos son los pasos que seguiremos a grandes rasgos:

1. Como estamos usando dos estructuras de datos, vamos a crear dos componentes (*origen* y *destino*) en el diseño de la asignación.
2. La transformación de un documento en otro se lleva a cabo usando un lenguaje de transformación[17] que tendremos que elegir.
3. Después tendremos que conectar los nodos de origen con los nodos de destino deseados. Estas conexiones son las que constituyen la asignación y determinan a qué nodo de destino está asignado cada nodo de origen.
4. Como resultado de la asignación obtenemos el documentos XML de destino, que es válido conforme al esquema de destino.
5. Por último, guardamos el archivo XML de salida.

Para más información sobre transformaciones de datos consulte el modelo abstracto que ve más abajo.

Modelo abstracto

El modelo abstracto que ve a continuación ilustra una transformación de datos del tutorial:



La asignación tiene un origen y un destino. El esquema de origen (**Books.xsd**) describe la estructura del archivo de instancia de origen (**Books.xml**). El esquema de destino (**Library.xsd**) describe la estructura del archivo de instancia de destino (**BooksOutput.xml**). Al conectar los nodos del componente de origen a los del componente de destino la asignación genera código de transformación en XSLT 3.0. El código de transformación lee código de **Books.xml** y escribe esos datos en **BooksOutput.xml**.

Archivos de origen y de destino

El fragmento de código siguiente contiene algunos de los datos de muestra de **Books.xml** que se usan como datos de origen.

```
<books>
   <book id="1">
      <author>Mark Twain</author>
      <title>The Adventures of Tom Sawyer</title>
      <category>Fiction</category>
      <year>1876</year>
   </book>
   <book id="2">
      <author>Franz Kafka</author>
      <title>The Metamorphosis</title>
      <category>Fiction</category>
      <year>1912</year>
   </book>
</books>
```

Y así es como queremos que queden los datos en el archivo de destino **BooksOutput.xml**:

```
<library>
   <last_updated>2015-06-02T16:26:55+02:00</last_updated>
   <publication>
      <id>1</id>
      <author>Mark Twain</author>
      <title>The Adventures of Tom Sawyer</title>
      <genre>Fiction</genre>
      <publish_year>1876</publish_year>
   </publication>
   <publication>
      <id>2</id>
      <author>Franz Kafka</author>
      <title>The Metamorphosis</title>
      <genre>Fiction</genre>
      <publish_year>1912</publish_year>
   </publication>
</library>
```

Lo que queremos es rellenar los elementos `<author>`, `<title>`, `<genre>` y `<publish_year>` del archivo de destino con el contenido de los elementos equivalentes del archivo de origen (`<author>`, `<title>`, `<category>`, `<year>`). El atributo `id` del archivo de origen se asigna al elemento `<id>` del archivo de destino. Por último, debemos rellenar el elemento `<last_updated>` del archivo de destino con la fecha y hora en que el archivo se actualizó por última vez.

Para llevar a cabo la transformación de datos siga los pasos que describimos en los apartados siguientes.

3.1.1 Crear y guardar diseños

En este apartado explicamos cómo crear un diseño nuevo, seleccionar un lenguaje de transformación, así como validar y guardar asignaciones.

Crear un diseño nuevo

Para poder llevar a cabo una transformación primero debe crear un diseño de asignación nuevo, algo que puede hacer de varias maneras:

- En el menú **Archivo** haga clic en el comando **Nuevo**.
- En la barra de herramientas haga clic en  .

Seleccionar el idioma de transformación

Según la edición de MapForce con la que trabaje tendrá disponibles distintos lenguajes de transformación[17]. En este tutorial hemos seleccionado XSLT3. Puede seleccionar este lenguaje de transformación de dos maneras diferentes:

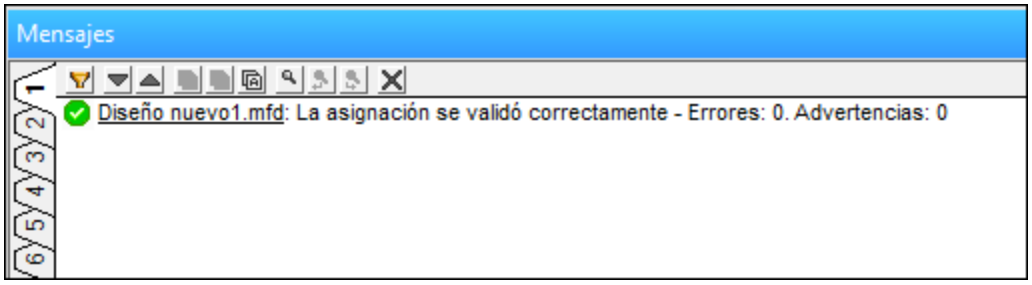
- En la barra de herramientas, haga clic en **XSLT3**.
- Abra el menú **Resultados** y haga clic en **XSLT 3.0**.

Validar y guardar el diseño

La validación es un paso opcional que permite detectar y corregir errores y advertencias potenciales antes de que se ejecute la asignación. Puede validar la asignación en cualquier momento. Para comprobar si una asignación es válidas tiene dos opciones:

- En el menú **Archivo**, haga clic en **Validar asignación**.
- En la barra de herramientas, haga clic en  .

En la ventana Mensajes aparece el resultado de la validación:



Para guardar la validación tiene dos opciones:

- En el menú **Archivo** haga clic en **Guardar**.
- En la barra de herramientas haga clic en  .

Hemos guardado la asignación creada en este tutorial como **`Tut1_OneToOne.mfd`**.

3.1.2 Agregar un componente de origen

En este punto queremos agregar un archivo XSD, que será la estructura del primer componente, y un archivo XML, que contenga los datos de ese componente. El archivo de origen **Books.xsd** se puede agregar a la asignación de varias maneras:

- En la barra de herramientas haga clic en  (Insertar archivo o esquema XML).
- En el menú **Insertar** haga clic en **Arc&hivo o esquema XML**.
- Arrastre **Books.xsd** desde el explorador de Windows hasta el área de asignación.

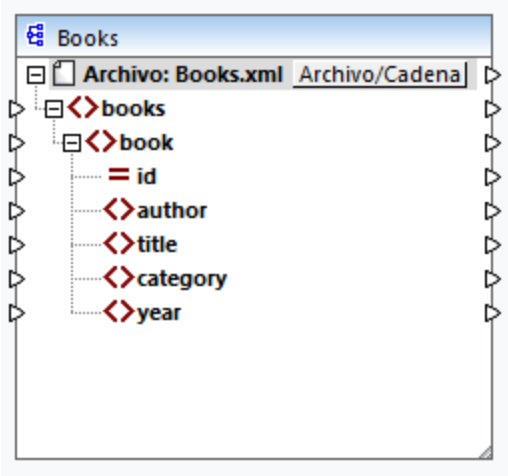
Si selecciona una de las dos primeras opciones, el cuadro de diálogo **Insertar un archivo de esquema XML** le sugerirá que elija entre un paquete de esquema estándar y un archivo remoto o local. En nuestros tutoriales usamos únicamente archivos locales. Para más información sobre cómo añadir archivos XML, consulte el apartado XML y esquemas XML 113.

Si se crea un componente a partir de un archivo XSD, la aplicación le pedirá que seleccione un archivo XML para usarlo como archivo de datos del componente. Si se crea un componente a partir de un archivo XML, el archivo XSD al que se hace referencia desde el archivo XML se usa para definir la estructura de los datos del componente. Si no existen referencias a ningún archivo XSD, MapForce le sugerirá generar uno para ese componente.

Dado que primero añadimos el archivo de esquema, MapForce sugiere que añadimos también un archivo XML de ejemplo. Haga clic en **Examinar** y navegue hasta **Books.xml**, que se encuentra en la misma carpeta. Ahora el archivo de origen contiene tanto un esquema como contenido.

Ver la estructura

Ahora que hemos añadido un archivo de origen al área de asignación, podemos ver su estructura. En MapForce esta estructura se conoce como componente de asignación o simplemente componente 30. Para ampliar elementos en el componente haga clic en el icono. También puede pulsar la tecla **+** del teclado numérico. En la imagen siguiente puede ver el componente de origen:



El nombre del encabezado (Books) solamente hace referencia al nombre del componente y no al nombre del esquema en el que se basa este archivo. Para ver el nombre del esquema y otras características del

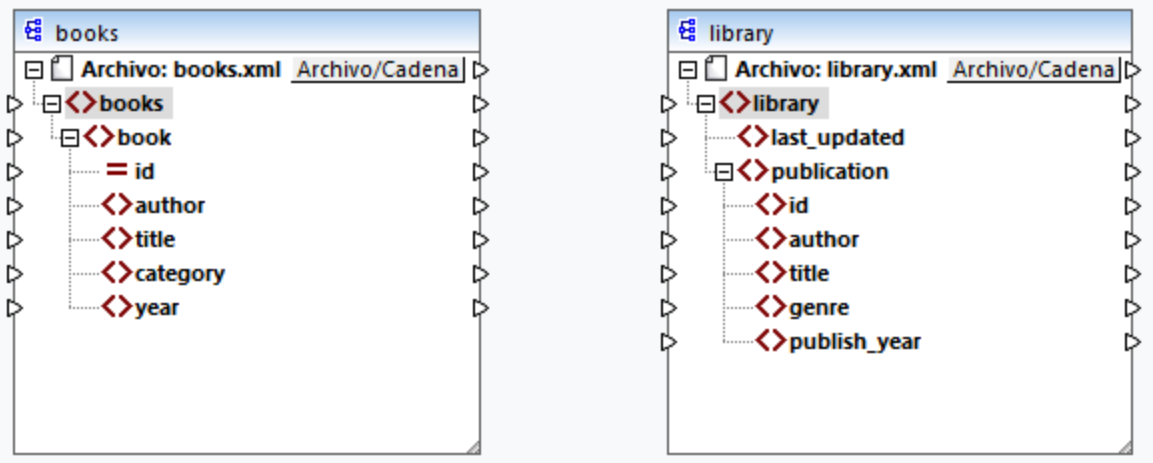
componente haga doble clic en el encabezado del componente. Esto abre el cuadro de diálogo Configuración del componente[114].

El nodo de nivel superior del componente (**`Archivo: Books.xml`**) representa el nombre del archivo de instancia XML. Los elementos XML de la estructura están representados por el icono . Los atributos XML están representados por el icono . Los triángulos pequeños que aparecen a ambos lados del componente representan la entrada de datos por la izquierda y la salida de datos por la derecha. En MapForce estos conectores se llaman respectivamente *conectores de entrada* y *conectores de salida*.

Para más información sobre componentes, conexiones, procedimientos y funciones generales, consulte el apartado Fundamentos de la asignación de datos[30].

3.1.3 Agregar un componente de destino

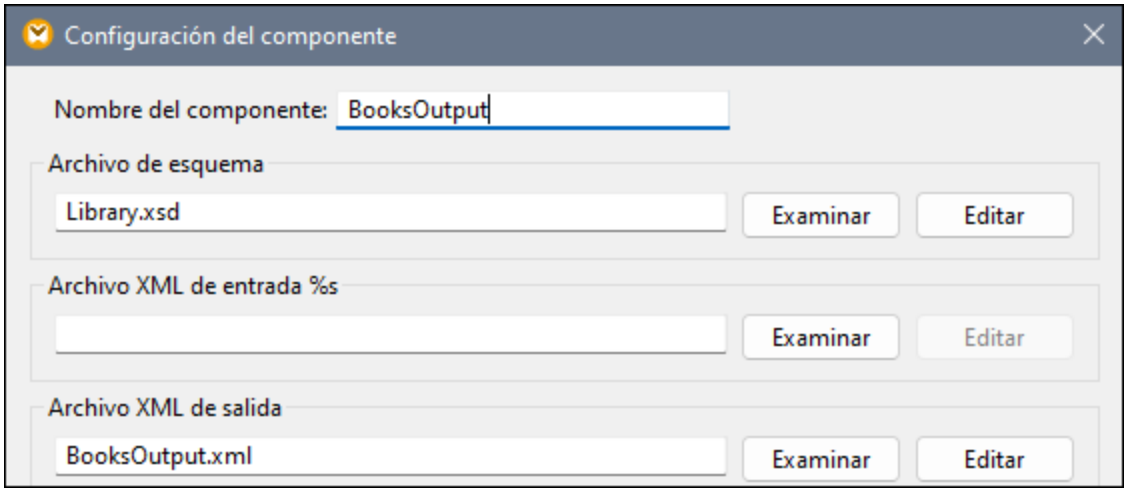
El paso siguiente es agregar un componente de destino y configurarlo. Agregue el archivo de destino **`Library.xsd`** a la asignación. Haga clic en **Omitir** cuando MapForce le pida que indique un archivo de instancia. En este punto la asignación tiene este aspecto:



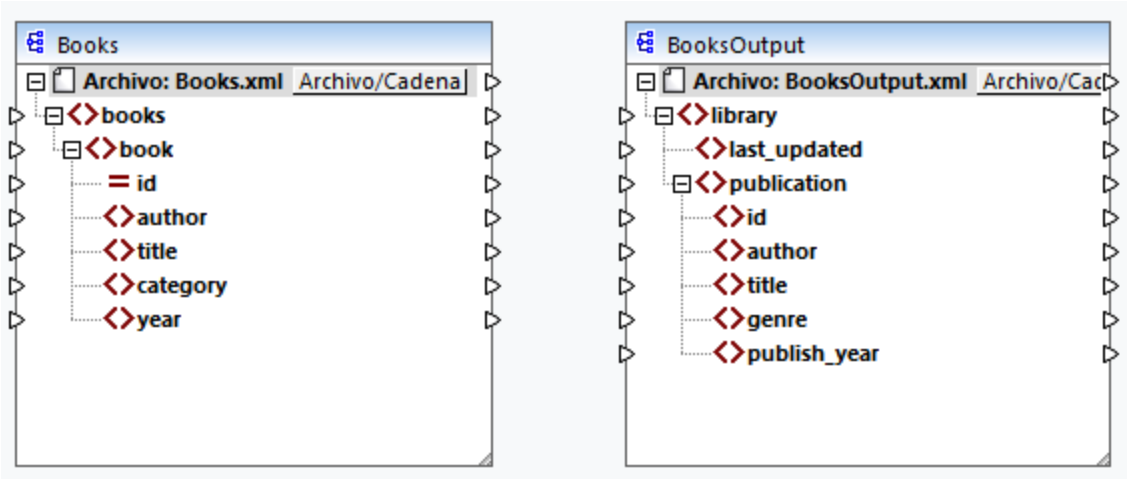
Como se puede ver, al abrir **`Library.xsd`**, este aparece como archivo XML en el componente. En realidad, lo que hace MapForce es crear una referencia al archivo XML llamada **`Library.xml`**, pero sin crear ningún archivo XML de verdad. Por tanto, el componente de destino ahora mismo tiene un esquema pero no tiene contenido.

Configurar componentes

Ahora cambiamos el nombre del componente de destino a **`BooksOutput.xml`**. Es decir que el archivo de salida se llamará **`BooksOutput.xml`**. Así evitamos confundirnos en los tutoriales siguientes, ya que vamos a usar un archivo aparte llamado **`Library.xml`**, que tiene contenido propio y se basa en el mismo esquema **`Library.xsd`**. Para cambiarle el nombre al archivo de destino haga doble clic en el encabezado del componente de destino. Se abre el cuadro de diálogo Configuración del componente[114] (*ver imagen siguiente*), donde vamos a cambiar el nombre del archivo de destino:



La asignación diseñar ahora tiene este aspecto:



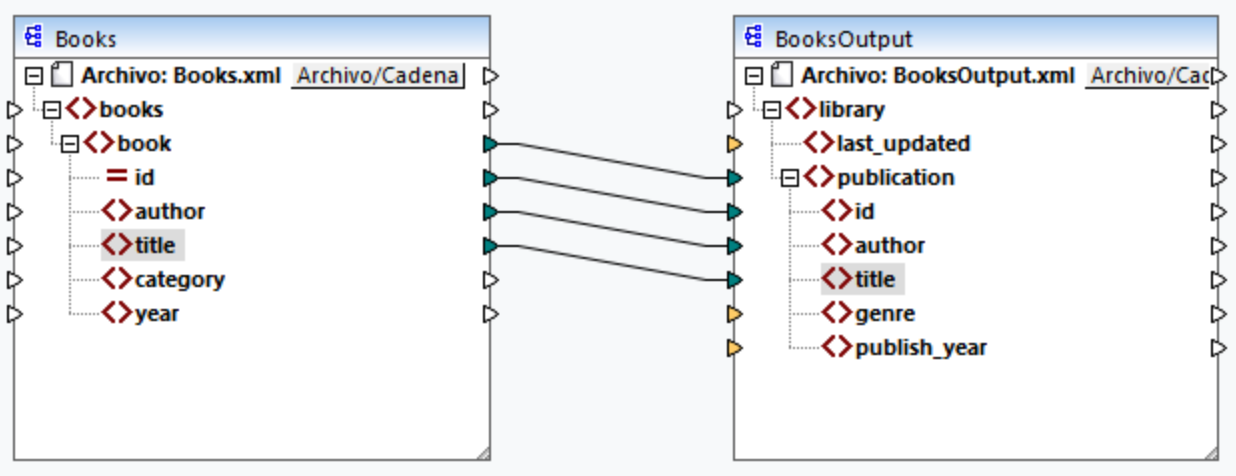
3.1.4 Conectar origen y destino

En este paso vamos a asignar los datos del archivo de origen al archivo de destino. También proporcionaremos información sobre la fecha y hora actuales con la función XPath `current-dateTime` 325.

Conexiones automáticas

Ahora vamos a crear una conexión entre el elemento <book> del componente de origen y el elemento <publication> del componente de destino. Para ello pulse y mantenga en el conector de salida (el pequeño triángulo) a la derecha del elemento <book> y arrastre la línea hasta el conector de entrada del elemento <publication> en el componente de destino. Con esta acción MapForce puede conectar automáticamente todos los nodos secundarios de <book> del archivo de origen a los nodos que tienen el mismo nombre en el archivo de destino. En este ejemplo se han creado cuatro conexiones de forma simultánea (*imagen siguiente*).

Esta acción se llama *Conectar automáticamente los secundarios equivalentes* [53] y se puede deshabilitar y también configurar.



Para habilitar o deshabilitar esta función:

- En la barra de herramientas haga clic en (**Conectar automáticamente los secundarios equivalentes**).
- En el menú **Conexión** haga clic en el comando **Conectar automáticamente los secundarios equivalentes**.

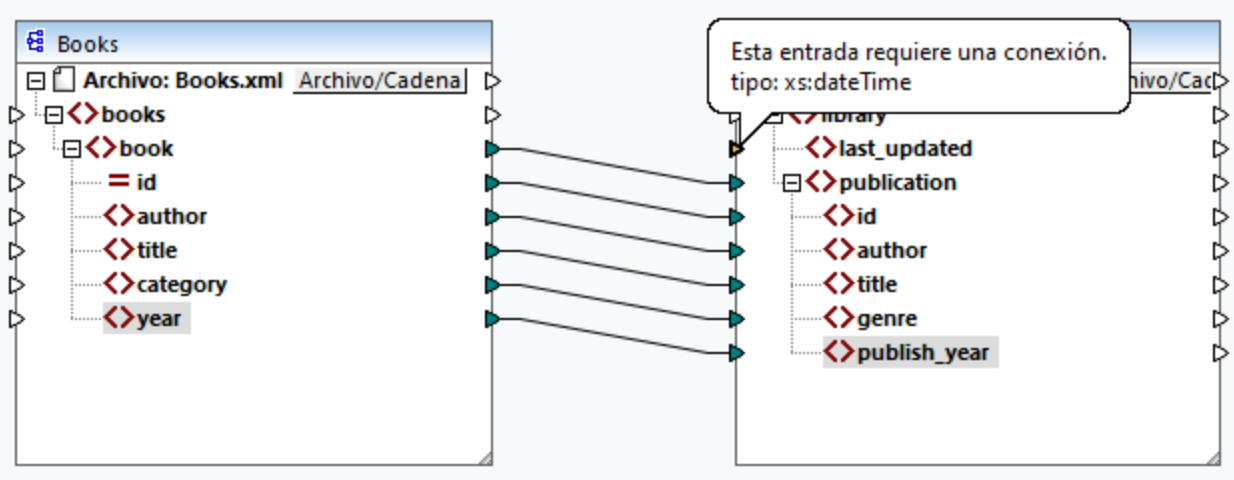
Conectar elementos obligatorios

Como puede ver en la imagen anterior, algunos de los conectores aparecen en color naranja, lo que indica que son obligatorios. Estos elementos son obligatorios porque así se han definido en el esquema del archivo. Para asegurarse de que el archivo XML de destino es válido e indicar valores para los elementos obligatorios:

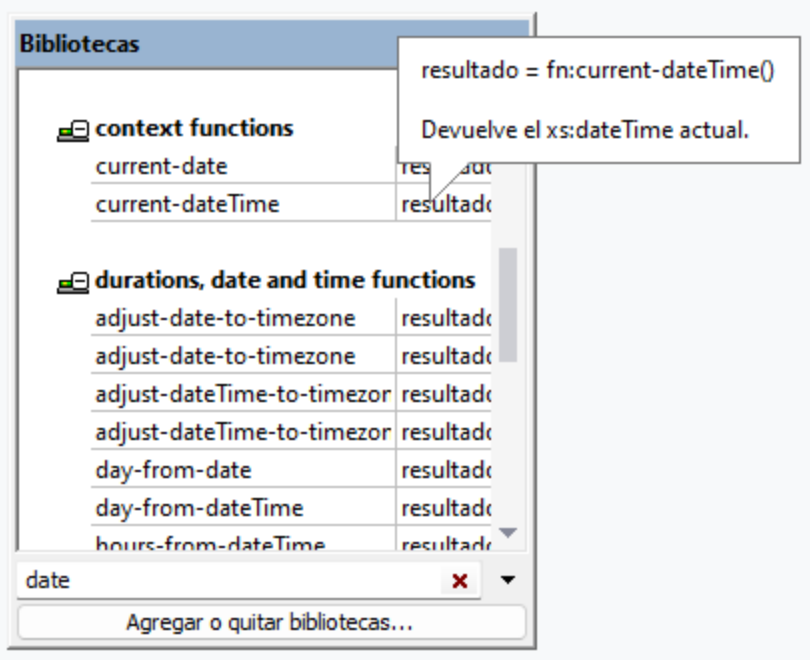
- Conecte el elemento `<category>` del archivo de origen con el elemento `<genre>` del componente de destino.
- Conecte el elemento `<year>` del archivo de origen con el elemento `<publish_year>` del componente de destino.

Agregar fecha y hora actuales

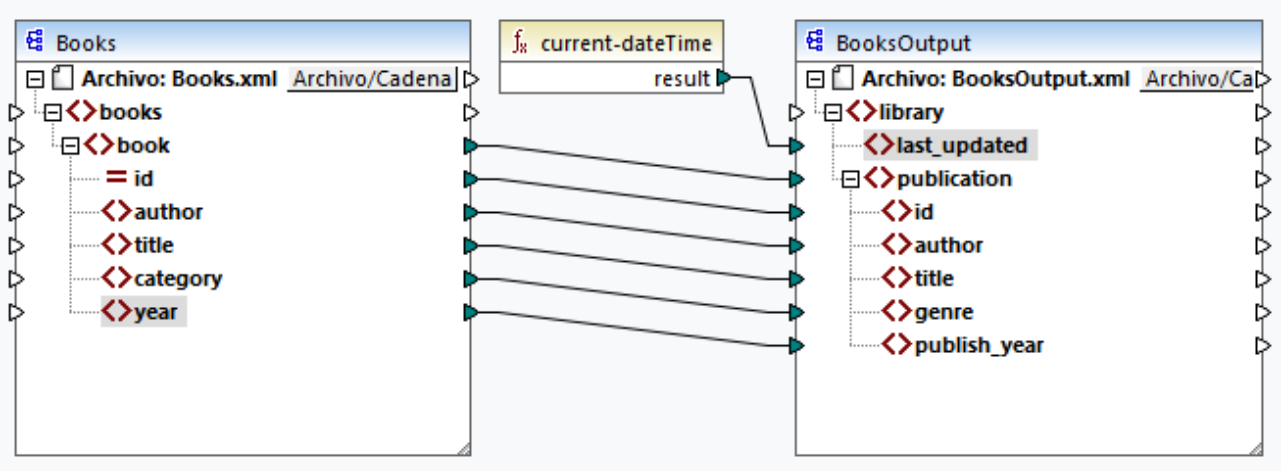
Por último debe indicar un valor para el elemento `<last_updated>`. Si pasa el cursor del ratón por encima de este conector de entrada puede ver que es un elemento de tipo `xs:dateTime` (*imagen siguiente*). Para poder ver esa información, pulse el botón de la barra de herramientas . Si pulsa el botón (**Mostrar tipos de datos**) de la barra de herramientas podrá ver el tipo de datos de los elementos en todo momento.



Para obtener la fecha y hora actuales puede usar la función **`current-dateTime`**. Para encontrarla empiece a teclearla en la caja de texto que hay en la parte inferior de la ventana Bibliotecas[21] (*imagen siguiente*). También puede hacer doble clic en un área vacía dentro del panel Asignación y empezar a teclear `current-date`.



Para agregar la función a la asignación basta con arrastrarla al panel Asignación. Después, conecte su conector de salida con el conector de entrada del elemento `<last_updated>` (*ver imagen siguiente*).



Ahora puede validar y guardar la asignación, como explicamos en Crear y guardar diseños[81].

3.1.5 Vista previa del resultado de la asignación

MapForce usa sus motores integrados para generar el resultado de la asignación y permite visualizar una vista previa del mismo en el panel Resultados (*ver imagen siguiente*).

Guardar resultados

Por defecto los archivos que aparecen en el panel Resultados no se guardan en disco. En su lugar, MapForce crea archivos temporales. Para guardar el resultado abra el panel Resultados y seleccione el comando de

menú **Resultados | Guardar archivo de salida** o haga clic en (**Guardar resultado generado**) en la barra de herramientas.

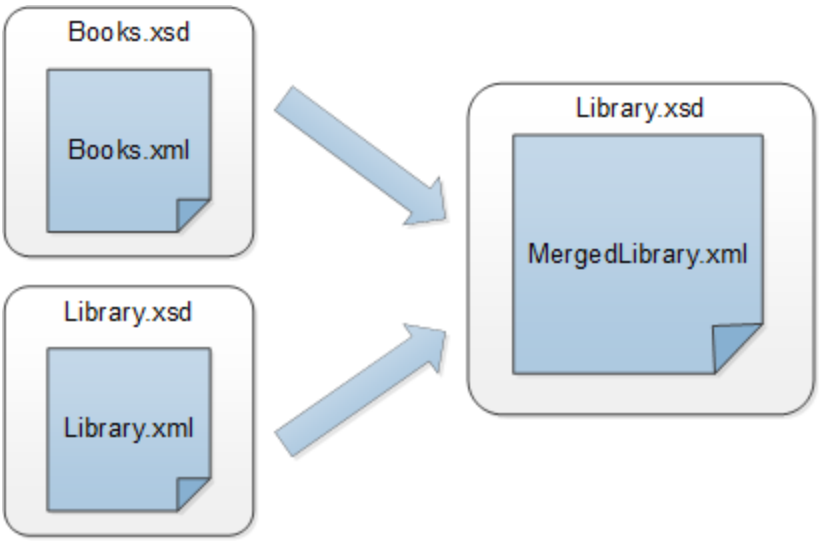
Para que MapForce escriba el resultado directamente en archivos finales en vez de temporales vaya a **Herramientas | Opciones | Generales** y marque la casilla de verificación *Escribir directamente en archivos de salida finales*. No recomendamos activar esta opción si está siguiendo el tutorial porque puede sobrescribir los archivos originales del tutorial sin querer.

Vista previa del código generado
También puede acceder a la vista previa del código XSLT generado que lleva a cabo la transformación. Para previsualizar el código, abra el panel XSLT3 que hay en la parte inferior de la ventana Asignación. Para generar el código XSLT3 y guardarlo en un archivo seleccione el elemento de menú **Archivo | Generar código en | XSLT 3.0**. Cuando la aplicación lo pida, seleccione la carpeta en la que quiere guardar el código generado. Una vez haya generado el código la carpeta de destino contendrá estos dos archivos:

1. Un archivo XSLT de transformación con el mismo nombre que el esquema de destino. Este es el formato del archivo de transformación: **`MappingMapTo<NombreArchivoDestino>.xslt`**.
2. Un archivo **`DoTransform.bat`**, que permite ejecutar la transformación XSLT con Altova RaptorXML Server desde la línea de comandos. Para ejecutar este comando necesita tener instalado RaptorXML.

3.2 Varios archivos de origen a un solo destino

En este tutorial aprenderá a combinar los datos de un archivo nuevo llamado **Library.xml** con los datos de **Books.xml**. El resultado será un archivo de destino llamado **MergedLibrary.xml** que contendrá los datos de los dos archivos de destino. El archivo de destino se basa en el esquema **Library.xsd**. Observe que los archivo de origen tienen distintos esquemas. Si los archivos de origen tuvieran el mismo esquema, podría combinar sus datos usando otro método, como explicamos en Varios archivos de origen a varios archivos de destino 103. En la imagen siguiente puede ver un modelo abstracto de la transformación de datos de este tutorial.



El fragmento de código siguiente contiene algunos de los datos de **Books.xml** que usaremos como origen de datos:

```
<books>
   <book id="1">
      <author>Mark Twain</author>
      <title>The Adventures of Tom Sawyer</title>
      <category>Fiction</category>
      <year>1876</year>
   </book>
</books>
```

El fragmento de código siguiente muestra un extracto de **Library.xml** que usaremos como segundo origen de datos:

```
<library>
   <publication>
      <id>5</id>
      <author>Alexandre Dumas</author>
      <title>The Three Musketeers</title>
      <genre>Fiction</genre>
      <publish_year>1844</publish_year>
   </publication>
</library>
```

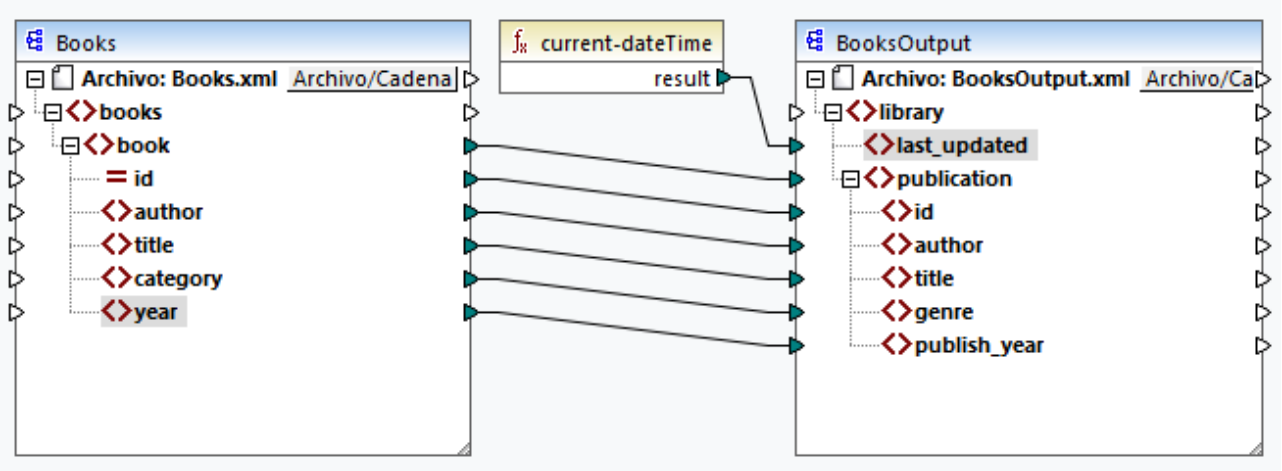
Este es el aspecto que queremos que tengan los datos combinados en el archivo de destino **MergedLibrary.xml**:

```
<library>
   <publication>
      <id>1</id>
      <author>Mark Twain</author>
      <title>The Adventures of Tom Sawyer</title>
      <genre>Fiction</genre>
      <publish_year>1876</publish_year>
   </publication>
   <publication>
      <id>5</id>
      <author>Alexandre Dumas</author>
      <title>The Three Musketeers</title>
      <genre>Fiction</genre>
      <publish_year>1844</publish_year>
   </publication>
</library>
```

Para llevar a cabo la transformación siga los pasos que describimos en los apartados siguientes.

3.2.1 Preparar el diseño de la asignación

El punto de partida de este tutorial es la asignación **Tut1_OneToOne.mfd** (*ver imagen siguiente*) que hemos diseñado en el primer tutorial[79]. Antes de realizar cualquier cambio en la asignación, asegúrese de guardar este diseño con un nuevo nombre en la carpeta **Tutoriales básicos**.

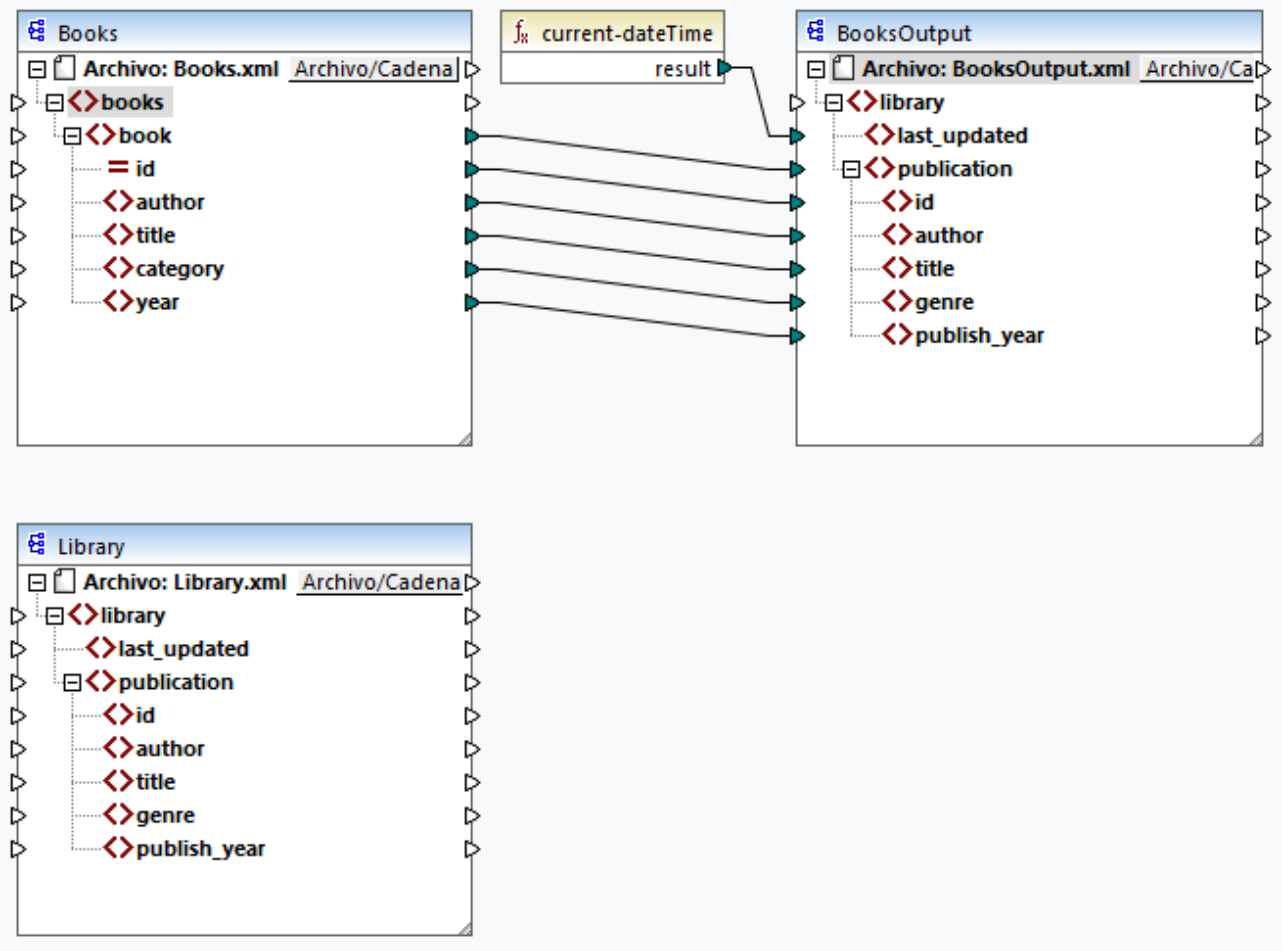


3.2.2 Agregar segundo archivo de origen

El paso siguiente consiste en agregar el segundo archivo de origen: Insertar el archivo **`Library.xml`** en la asignación. Ya que se hace referencia el esquema (**`Library.xsd`**) en este archivo XML, no es necesario añadir el archivo del esquema en otro paso separado. Para comprobar si la referencia al esquema es correcta, abra la configuración de los componentes[114].

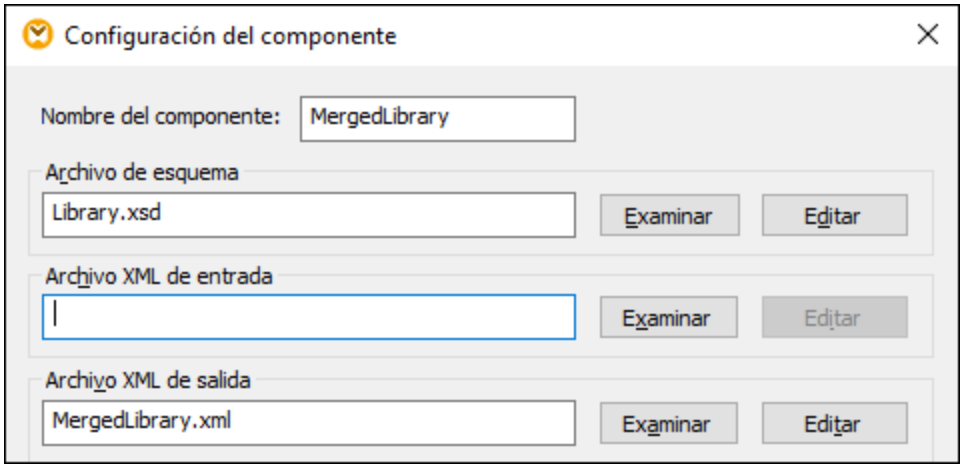
Ahora pulse y mantenga pulsado el encabezado del componente nuevo y póngalo bajo el componente `Books`. Puede mover los componentes en cualquier momento y dirección. Sin embargo, si coloca los componentes de origen a la izquierda de los de destino la asignación será más fácil de leer y de entender. Esta es la convención que usan en todas las asignaciones de esta documentación, así como los archivos de asignación de ejemplo que se instalan con MapForce.

En este punto la asignación tiene este aspecto:



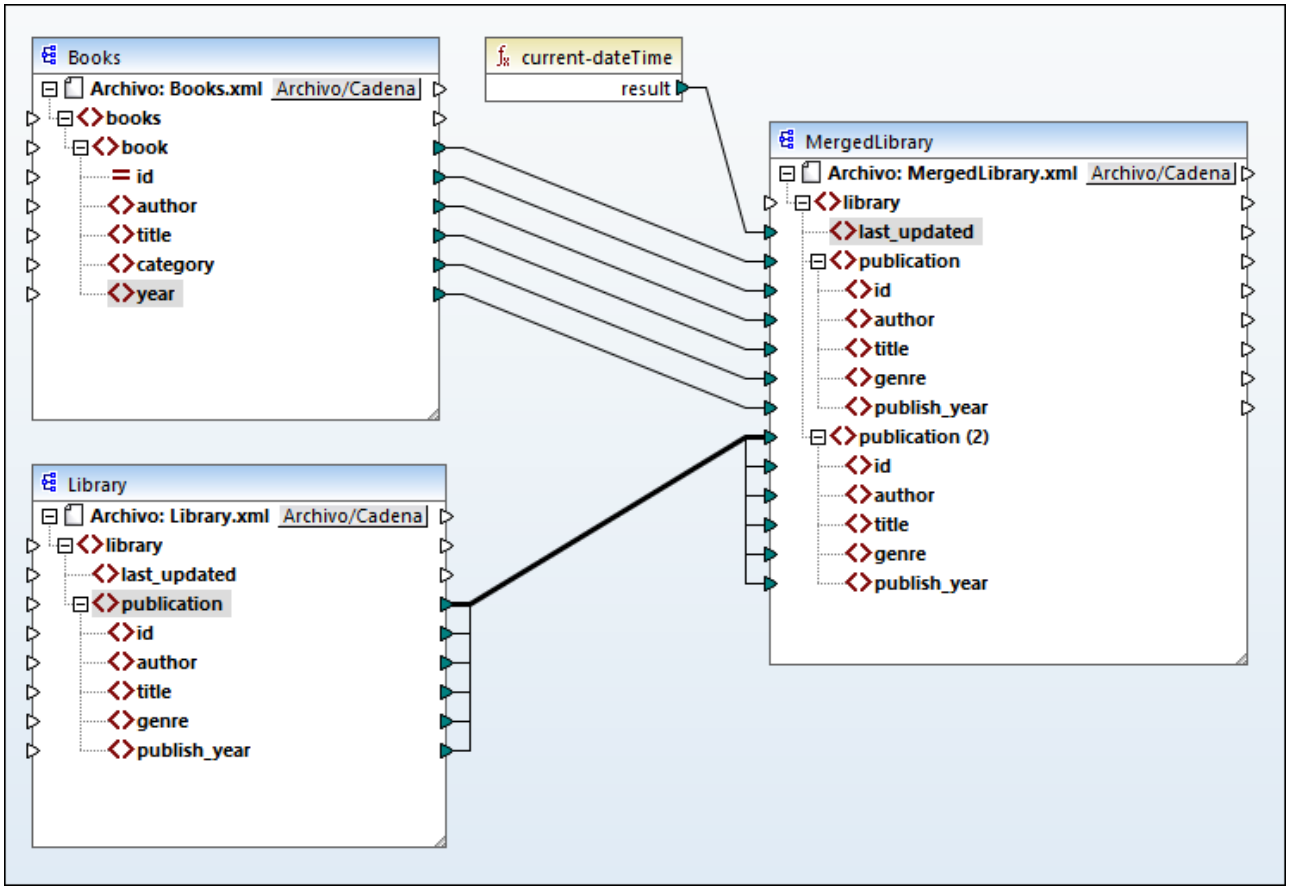
3.2.3 Configurar componentes de destino

En este punto la asignación tiene dos componentes de origen (`Books` y `Library`) y un componente de destino (`BooksOutput`). Para asegurar la coherencia y evitar malentendidos, tendremos que cambiar la configuración del componente `BookOutput`. Haga doble clic en el encabezado del componente. Esto abre el cuadro de diálogo Configuración del componente 114. Configura las opciones tal y como se explica a continuación.



3.2.4 Conectar varios orígenes a un destino

El último paso consiste en conectar el segundo componente de origen (`Library`) con el componente de destino (`MergedLibrary`). Para ello, conecte el elemento `<publication>` en **`Library.xml`** con el elemento `<publication>` en **`MergedLibrary.xml`**. Puesto que el conector de entrada de destino ya tiene una conexión, MapForce le pedirá que reemplace o duplique la entrada. En este tutorial el objetivo es asignar los datos de dos componentes de origen a uno de destino. Por tanto debemos hacer clic en **Agregar un duplicado de entrada**. Con esta acción se configura el componente de destino de forma que acepte también los datos del segundo componente de origen. La asignación ahora tiene este aspecto:



En la imagen anterior se observa que el elemento `publication` del componente de destino se ha duplicado. El nodo nuevo `publication(2)` aceptará los datos de **`Library.xml`**. Hay que recalcar que aunque el nombre de este nodo aparezca como `publication(2)` en la asignación, su nombre en el archivo XML de destino será `publication`, que es lo que queremos en este caso.

Conexión de copia total

Los elementos secundarios de ambos elementos `publication` en los dos componentes, es decir, tanto en `Library` como en `MergedLibrary`, tienen los mismos nombres y tipos de datos. Por tanto, estos elementos están conectados con una línea gruesa. Este tipo de conexión se llama *conexión de copia total* [55] y facilita la comprensión de la asignación.

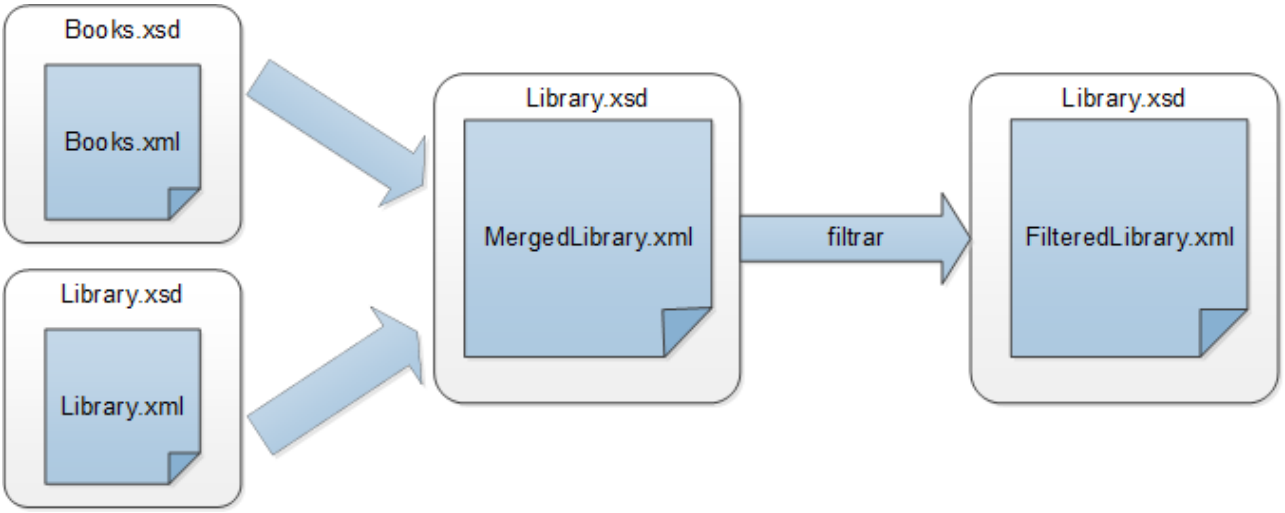
Vista previa del resultado

Abra la pestaña Resultados para ver el resultado. Observará que los datos de **`Books.xml`** y de **`Library.xml`** ahora se han combinado en el archivo nuevo **`MergedLibrary.xml`**. Hemos guardado el diseño de asignación de este tutorial como **`Tut2_MultipleToOne.mfd`**. Esta es la asignación que usaremos como punto de partida en el tutorial siguiente [94].

3.3 Asignación encadenada

Sitio web de Altova: Vídeo de tutorial sobre las asignaciones encadenadas

En este tutorial aprenderá a trabajar con varios componentes de destino. El objetivo de este tutorial es combinar los datos de dos archivos de origen en un único archivo de destino para después filtrar los datos resultantes y pasarlos a un segundo archivo de destino. En la imagen siguiente puede ver un modelo abstracto de la transformación de datos de este tutorial.



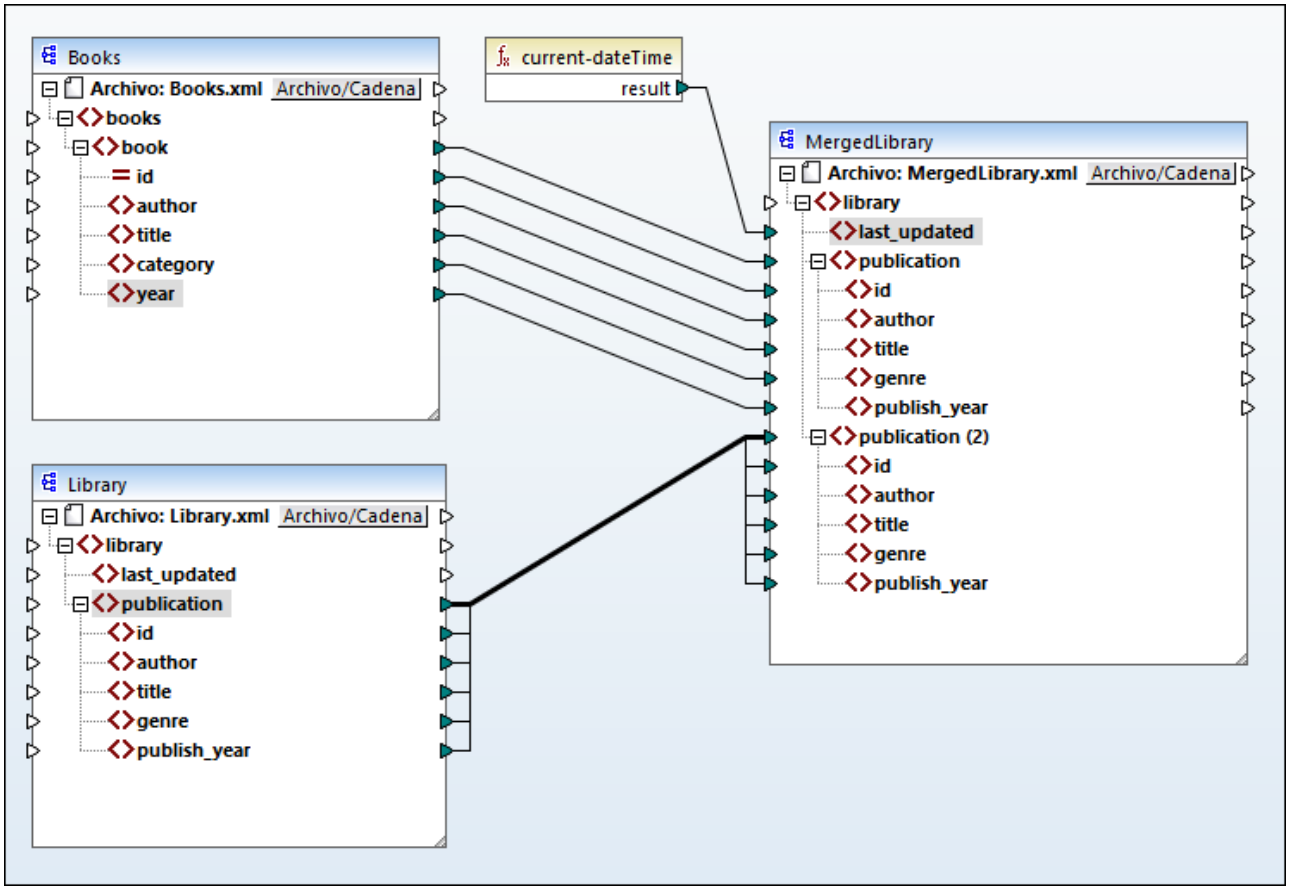
En este diagrama puede ver que primero se combinan los datos de los dos archivos de origen (**Books.xml** y **Library.xml**) en un único archivo de destino llamado **MergedLibrary.xml**. Después los datos se transforman con una función de filtro y se pasan al siguiente componente, que es **FilteredLibrary.xml**. Recuerde que **FilteredLibrary.xml** se basa en el esquema **Library.xsd**. El componente intermedio actúa como entrada de datos y también como salida. En MapForce esta técnica se conoce como *asignaciones encadenadas*. En las asignaciones encadenadas, puede previsualizar y guardar uno o varios resultados intermedios de la asignación (en nuestro caso es **MergedLibrary.xml**) y el resultado del último componente de destino (en nuestro caso es **FilteredLibrary.xml**).

Para llevar a cabo la transformación siga los pasos que describimos en los apartados siguientes.

Para ver otro ejemplo de una asignación encadenada, véase el tutorial en vídeo anexado en la parte superior de la página.

3.3.1 Preparar el diseño de la asignación

El punto de partida de este tutorial es la asignación **Tut2_MultipleToOne.mfd** (*imagen siguiente*). Esta es la asignación que diseñamos en el tutorial anterior[89]. Antes de realizar cualquier cambio en la asignación, asegúrese de guardar este diseño con un nuevo nombre a la carpeta **Tutoriales básicos**.

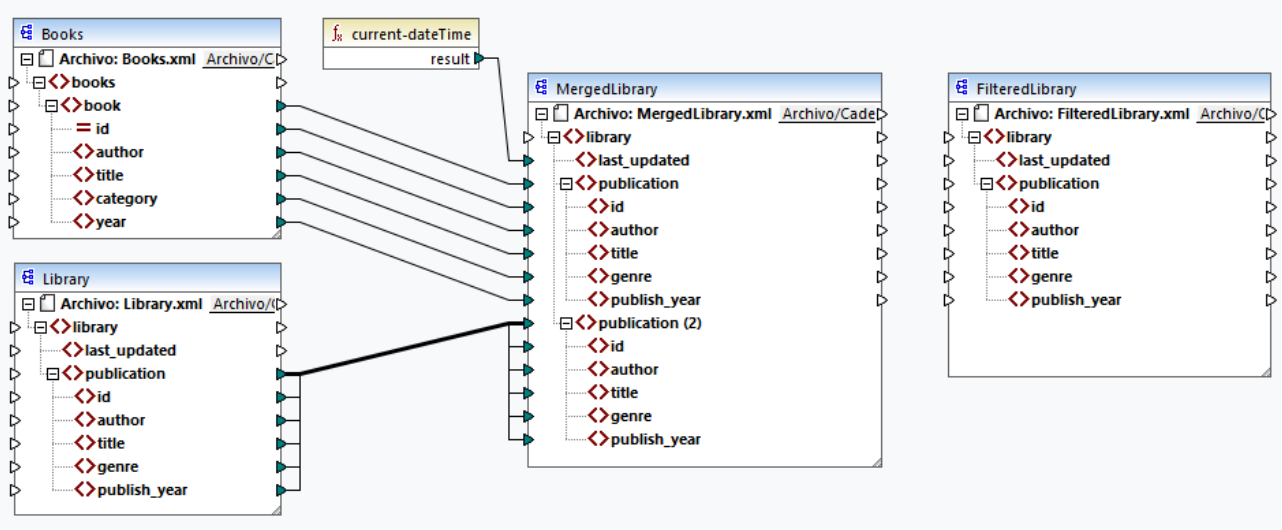


3.3.2 Configurar el segundo archivo de destino

Ahora tenemos que añadir y configurar el segundo archivo de destino que contiene solamente un subconjunto de los datos de `publicación` provenientes de **`MergedLibrary.xml`**.

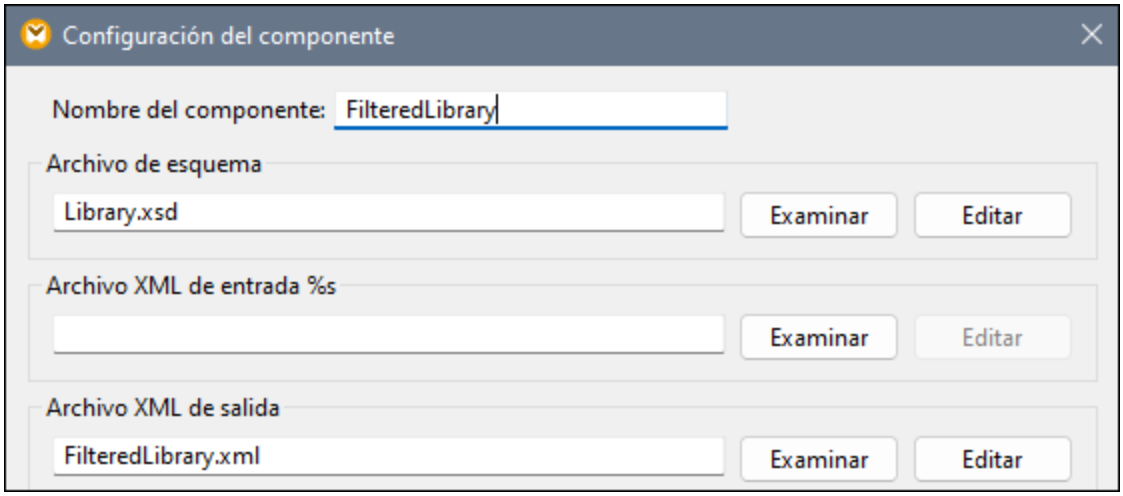
Agregar el segundo componente de destino

Agregue el archivo **`Library.xsd`** a la asignación y haga clic en **Omitir** cuando MapForce le pida que indique un archivo de instancia de ejemplo. El segundo componente de destino tiene estructura pero no contenido. Más adelante asignaremos los datos filtrados a este archivo de destino. El diseño de la asignación ahora tiene este aspecto:



Configurar el segundo componente de destino

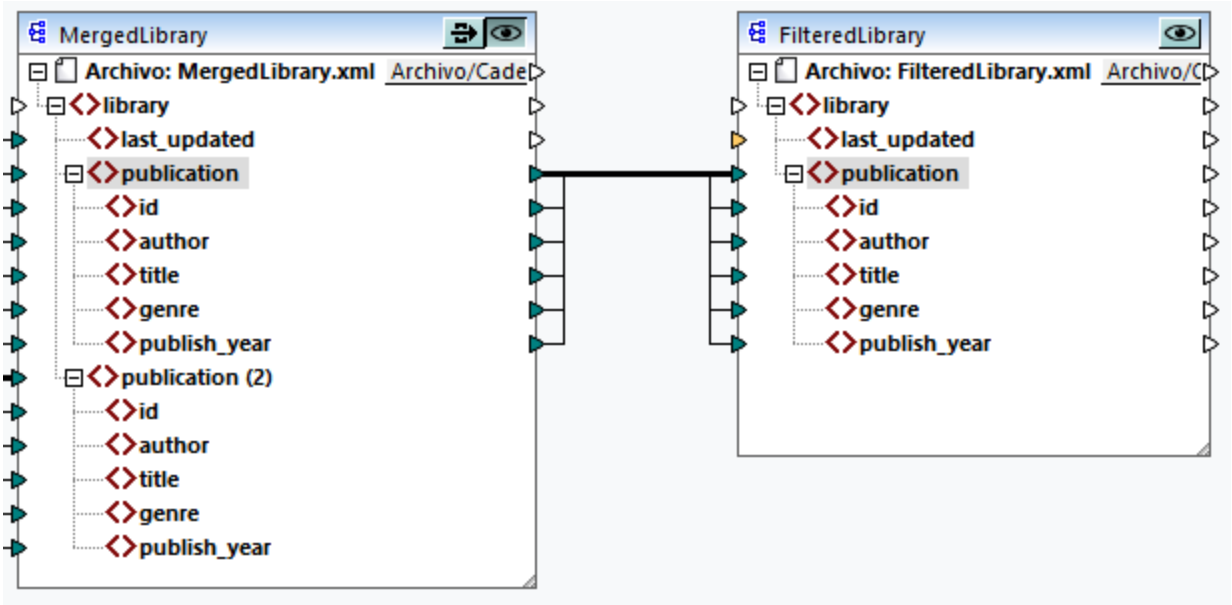
Como se ve más arriba, la asignación ahora tiene dos componente de origen (`Books` and `Library`) y dos componente de destino (`MergedLibrary` y `Library`). Para evitar malentendidos vamos a cambiar el nombre del componente que acabamos de añadir, `FilteredLibrary`. Para ello haga doble clic en el encabezado del componente que hay más a la derecha y edite su configuración[114] como sigue:



3.3.3 Conectar componentes de destino

El paso siguiente es asignar el elemento `publication` de `MergedLibrary` al elemento `publication` de `FilteredLibrary`. Al conectar el conector de salida de `MergedLibrary` con el conector de entrada de `FilteredLibrary`,MapForce le informará de que ha creado varios componentes de destino en la asignación. Observe que ahora hay botones nuevos disponibles en la esquina superior derecha de los dos componente de

destino: (**Preview**) y (**Pass-through**). Usaremos y explicaremos estos botones en los pasos siguientes.

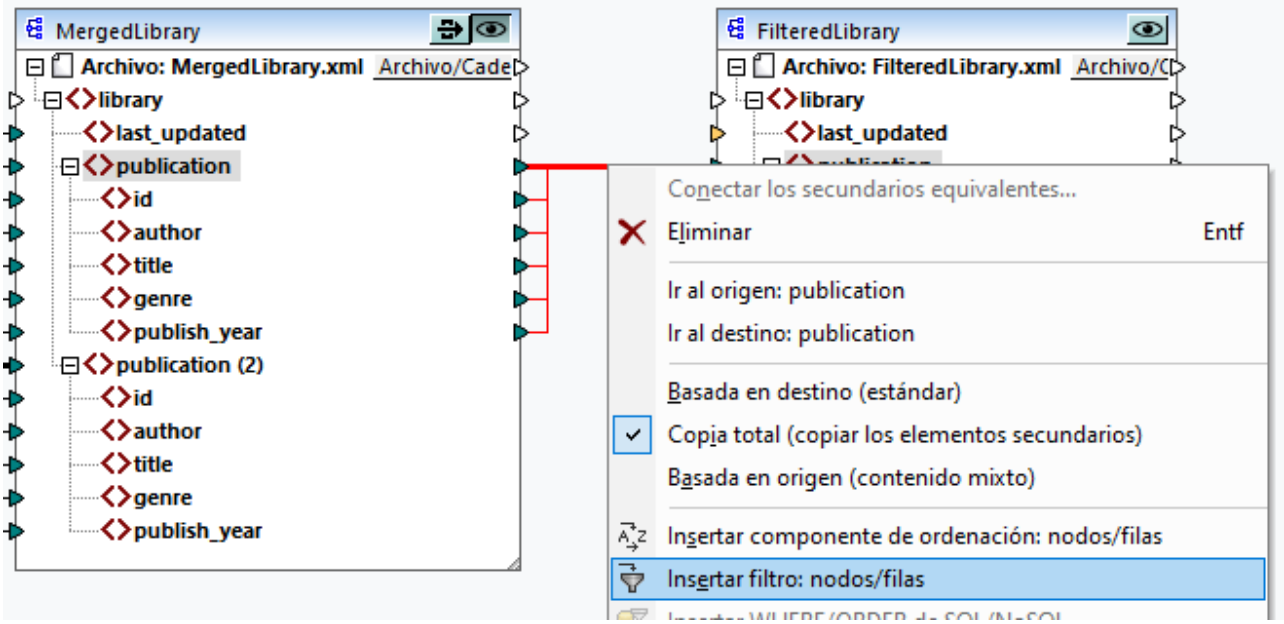


3.3.4 Filtrar datos

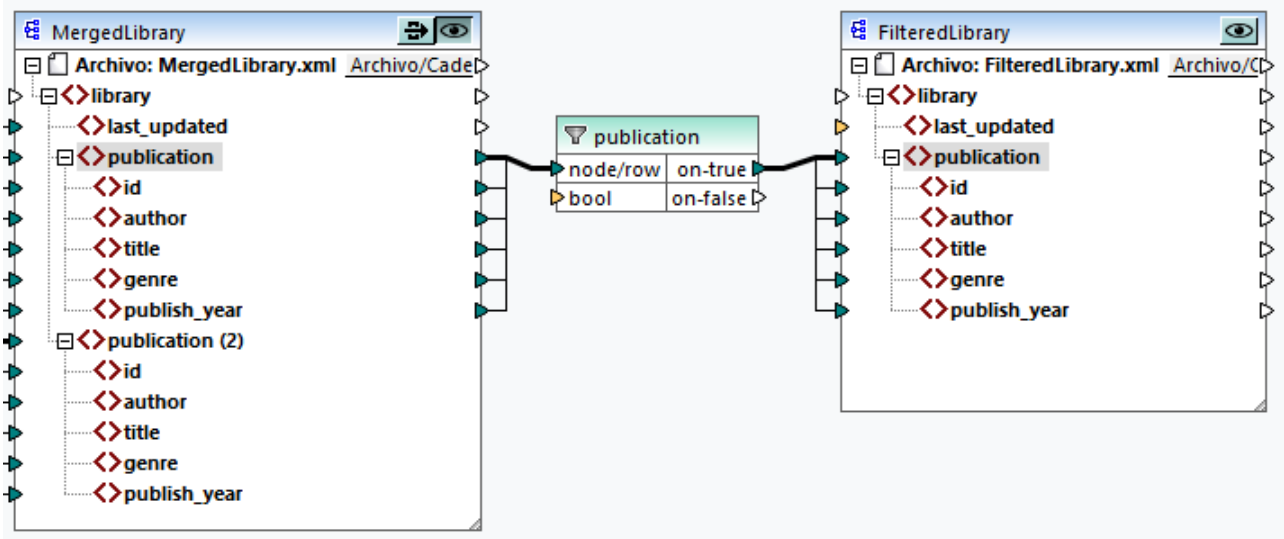
En este paso vamos a filtrar los datos de `MergedLibrary` para que solamente pasen al componente `FilteredLibrary` los libros publicados a partir de 1900. Para ello usaremos un componente Filtro.

Agregar un filtro

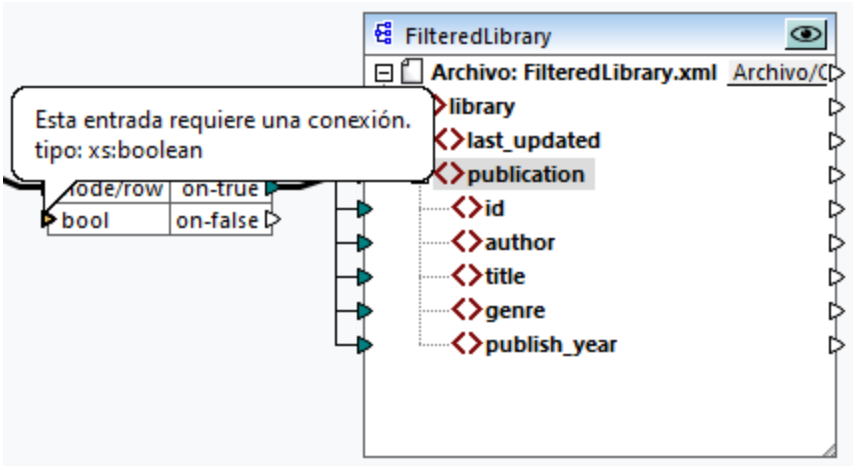
Para agregar un filtro haga clic con el botón derecho en la conexión entre `MergedLibrary` y `FilteredLibrary` y seleccione **Insertar filtro**: **nodos/filas** en el menú contextual (*ver imagen siguiente*).



Ahora hay un componente de filtro en la asignación (*ver imagen siguiente*).



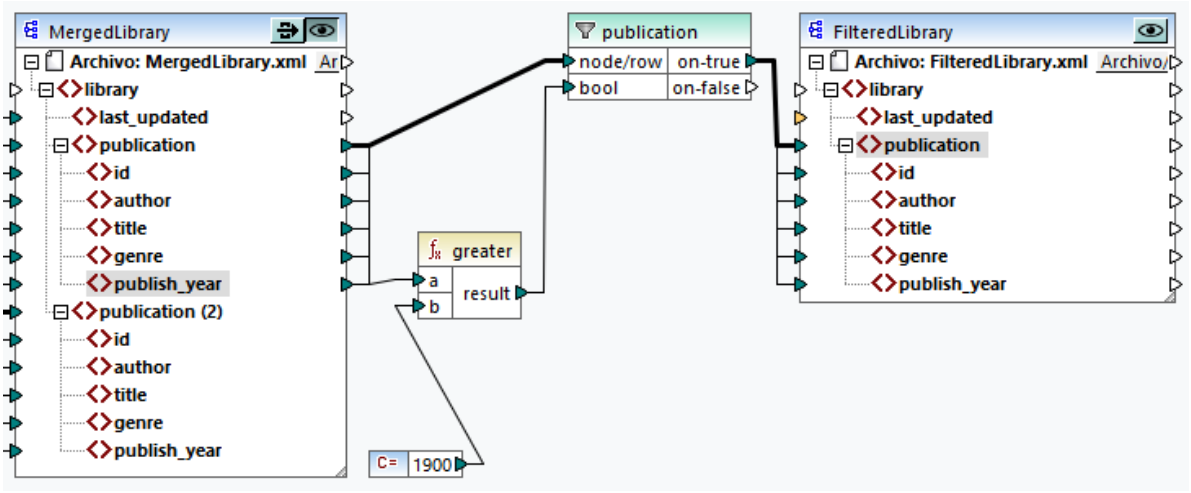
En la imagen anterior, el conector de entrada `bool` aparece en naranja, lo que significa que esta entrada es obligatoria. Si pasa el cursor del ratón por encima del conector verá que es necesario introducir un elemento de entrada de tipo `xs:boolean` (*ver imagen siguiente*). Para ver la información rápida habilítela primero haciendo clic en  en la barra de herramientas (**Mostrar información rápida**).



Sólo los libros publicados a partir de 1900

El componente de filtro necesita una condición que devuelva `true` o `false`. Si la condición booleana devuelve `true`, los datos de la secuencia `publication` actual se copian en el destino. Si la condición devuelve `false` no se copian datos. En este tutorial la condición asigna únicamente los libros que se publicaron a partir de 1900. Siga estos pasos para crear la condición:

1. Pulse **Constante** en la barra de herramientas y escriba *1900* en la barra de texto. Seleccione *Número* como tipo.
2. Agregue la función **`greater`** a la asignación.
3. Trace las conexiones de asignación a y desde la función **`greater`** como se ve en la imagen siguiente. La función **`greater`** comparará el valor del elemento `publish_year` de cada publicación con el valor de la constante. Sólo se asignarán al destino los registros de publicación cuyo año de publicación sea superior a 1900.



3.3.5 Previsualizar y guardar resultados

Ahora estamos listos para acceder a la vista previa y guardar el resultado de los componentes de destino. Si existen varios componentes de destino en una misma asignación, cada uno de estos componentes de destino tiene un botón  (**Vista previa**). La vista previa no se puede habilitar para más de un componente al mismo tiempo. Los botones **Vista previa** permiten ver el resultado intermedio de la asignación (en nuestro ejemplo, los datos asignados al componente `MergedLibrary`), así como el resultado final de la asignación encadenada (los datos asignados al componente `FilteredLibrary`). El componente `MergedLibrary` también tiene un botón  (**Paso a través**). El botón **Paso a través** controla cómo se generará el resultado (*ver detalles más abajo*).

Previsualizar y guardar resultados intermedios y finales
Para visualizar y guardar los resultados de los componentes `MergedLibrary` y `FilteredLibrary`, siga estos pasos:

1. Haga clic en el botón **Paso a través** del componente `MergedLibrary`.
2. Asegúrese de que el botón **Vista previa** del componente `FilteredLibrary` también está activado.
3. Abra la pestaña Resultados. Los botones **Atrás** y **Adelante** le permiten pasar de un resultado a otro.
4. Cambie al resultado que desea guardar en un archivo y haga clic en **Guardar archivo de salida** en la barra de herramientas. Si desea guardar ambos resultados, haga clic en el botón **Guardar todos los resultados generados** de la barra de herramientas.

Cuando la función del paso a través está activo, el campo *Archivo XML de entrada* del componente intermedio se desactiva automáticamente. Esto se debe a que los resultados generados al previsualizar la primera parte de la asignación entre los orígenes (`Books` y `Library`) y el primer destino (`MergedLibrary`) se utilizan por defecto como entrada cuando previsualiza la segunda parte de la asignación entre el primer destino (`MergedLibrary`) y el segundo destino (`FilteredLibrary`).

Previsualizar y guardar resultados intermedios
Los componentes intermedios no pueden previsualizarse cuando su botón de paso a través está activo. El botón de vista previa del componente intermedio se desactiva automáticamente porque no tiene sentido previsualizar y dejar pasar datos al mismo tiempo. Para visualizar y guardar sólo los resultados del componente intermedio (`MergedLibrary`) siga estos pasos:

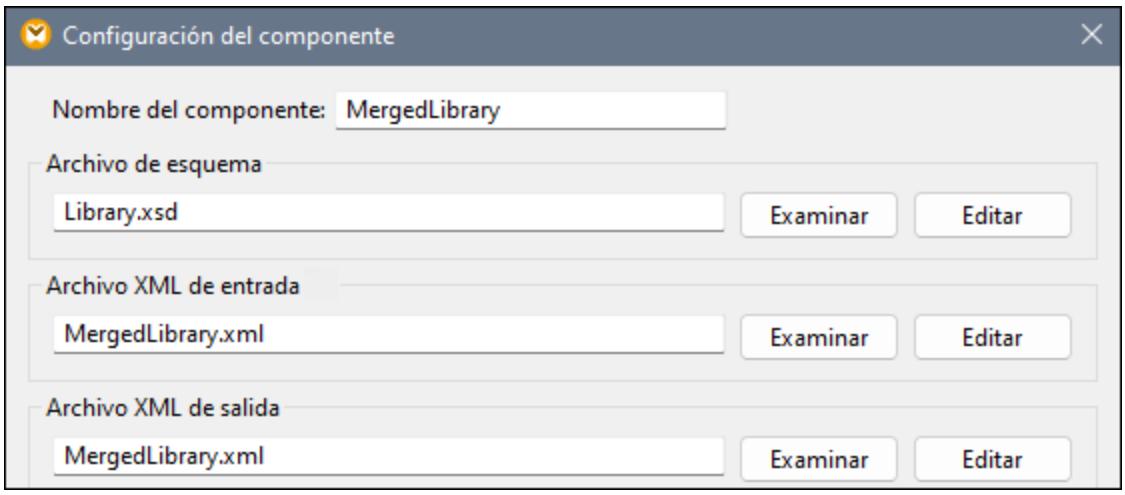
1. Desactive el botón **Paso a través** del componente `MergedLibrary` si estaba activado anteriormente.
2. Haga clic en el botón **Vista previa** del componente intermedio `MergedLibrary`.
3. Abra la pestaña Resultados.
4. Haga clic en el botón **Guardar archivo de salida** de la barra de herramientas para guardar los resultados en un archivo.

Previsualizar y guardar resultados finales
Para visualizar y guardar sólo los datos asignados del componente intermedio al segundo componente de destino, siga estos pasos:

1. Desactive el botón **Paso a través** del componente `MergedLibrary` si estaba activado anteriormente.
2. Haga doble clic en el encabezado del componente `MergedLibrary`.
3. Asegúrese de introducir el mismo nombre de archivo en el campo *Archivo XML de entrada* que en el campo *Archivo XML de salida* (*imagen siguiente*). Al desactivar el botón **Paso a través** puede elegir qué archivo de entrada debe leer el componente intermedio. En la mayoría de los casos, el archivo de entrada debería ser el mismo archivo que se definió en el campo *Archivo XML de salida*. Lo más

razonable es que el componente intermedio reciba un archivo, lo procese y lo reenvíe a la siguiente asignación, en lugar de buscar un nombre de archivo distinto.

Es importante tener el mismo archivo de entrada y salida para el componente intermedio cuando se desactiva el botón **Paso a través** del componente intermedio. Esto asegura que los resultados generados al previsualizar la primera parte de la asignación entre los orígenes (`Books` y `Library`) y el primer destino (`MergedLibrary`) se utilizan como entrada cuando previsualiza la segunda parte de la asignación entre el primer destino (`MergedLibrary`) y el segundo destino (`FilteredLibrary`). Si ejecuta su asignación con MapForce Server (*ediciones Professional y Enterprise*) o con código generado, los mismos nombres de los archivos de entrada y salida del componente intermedio garantizan que no se rompa la cadena de asignación. Tenga en cuenta que si en el componente intermedio el archivo de entrada y de salida tienen nombres *distintos* (con el botón **Paso a través** inactivo), pueden producirse errores en MapForce, en el código generado o en la ejecución con MapForce Server.



4. Haga clic en el botón **Vista previa** del componente `FilteredLibrary`.
5. Abra la pestaña Resultados.
6. Haga clic en el botón **Guardar archivo de salida** de la barra de herramientas para guardar los resultados en un archivo.

Importante

- La característica paso a través solamente está disponible para componentes basados en archivos, como componentes XML, CSV y de texto. Los componentes de base de datos (*ediciones Professional y Enterprise*) pueden ser componentes intermedios pero no disponen del botón Paso a través.
- Cuando MapForce ejecuta la asignación, la configuración de la opción *Escribir directamente en archivos de salida finales* (que se configura desde **Herramientas | Opciones | Generales**[472]) determina si los archivos de salida se deben guardar como archivos temporales o físicos. Recuerde que esto solo es válido cuando se consulta una vista previa en MapForce. Si la asignación se ejecuta con MapForce o por medio de código generado, se producirían archivos en cada etapa de la cadena de asignación.

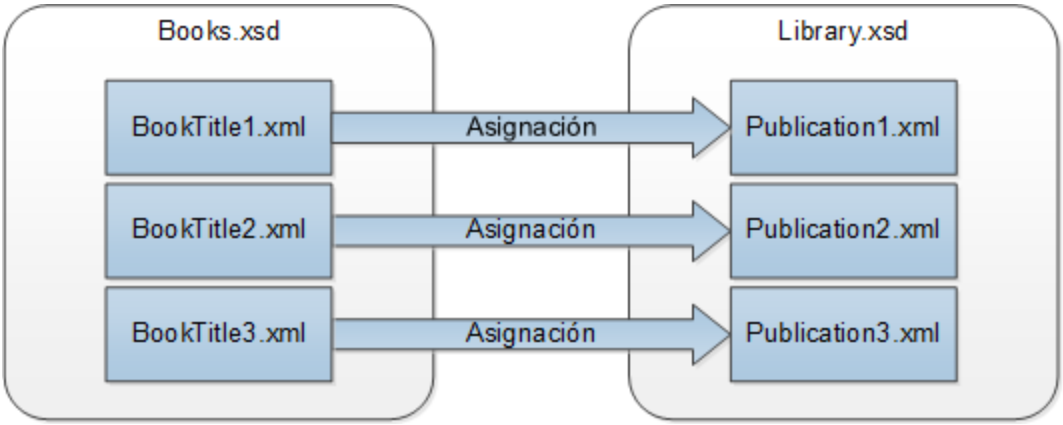
Con este paso terminamos de diseñar la asignación encadenada que produce dos archivos de salida. Hemos guardado el diseño de asignación de este tutorial como **`Tut3_ChainedMapping.mfd`**.

3.4 Varios archivos de origen a varios archivos de destino

En este tutorial aprenderá a asignar los datos de varios archivos de origen a varios archivos de destino en una sola transformación. Para que vea cómo funciona este proceso vamos a crear una asignación con estos objetivos:

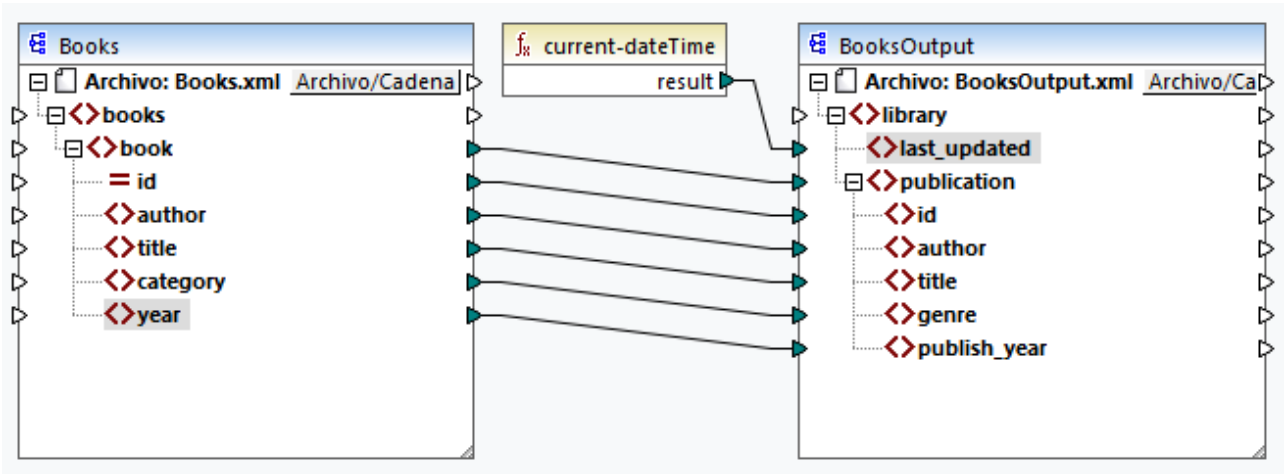
1. Leer datos de varios archivos XML ubicados en un mismo directorio. Estos archivos se basan en el mismo esquema.
2. Generar por cada archivo XML de origen un archivo XML de destino. Los archivos de destino se basarán en un nuevo esquema de destino.

En la imagen siguiente puede ver un modelo abstracto de la transformación de datos de este tutorial:



Resumen general

El punto de partida de este tutorial es la asignación **`Tut1_OneToOne.mfd`** del primer tutorial[79] (*ver imagen siguiente*).



Modificar el componente de origen
Modificaremos la configuración del componente de origen para que lea datos de distintos archivos de origen: **BookTitle1.xml**, **BookTitle2.xml** y **BookTitle3.xml**. Cada uno de los tres archivos se basa en **Books.xsd** y guarda un registro de libro (*ver a continuación*).

BookTitle1.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<books xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="Books.xsd">
   <book id="1">
      <author>Mark Twain</author>
      <title>The Adventures of Tom Sawyer</title>
      <category>Fiction</category>
      <year>1876</year>
   </book>
</books>
```

BookTitle2.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<books xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="Books.xsd">
   <book id="2">
      <author>Franz Kafka</author>
      <title>The Metamorphosis</title>
      <category>Fiction</category>
      <year>1912</year>
   </book>
</books>
```

BookTitle3.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<books xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="Books.xsd">
   <book id="3">
      <author>Herman Melville</author>
      <title>Moby Dick</title>
      <category>Fiction</category>
      <year>1851</year>
   </book>
</books>
```

Modificar el componente de destino
También modificaremos la configuración del componente de origen para que escriba datos a varios archivos de destino. Los archivos de destino se basan en el mismo esquema llamado **Library.xsd**. Los archivos de destino generados se llamarán **Publication1.xml**, **Publication2.xml** y **Publication3.xml** (*ver los extractos de código a continuación*).

Publication1.xml

```
<library>
   <publication>
```



```
            <id>1</id>
            <author>Mark Twain</author>
            <title>The Adventures of Tom Sawyer</title>
            <genre>Fiction</genre>
            <publish_year>1876</publish_year>
         </publication>
      </library>
```

Publication2.xml

```
      <library>
         <publication>
            <id>2</id>
            <author>Franz Kafka</author>
            <title>The Metamorphosis</title>
            <genre>Fiction</genre>
            <publish_year>1912</publish_year>
         </publication>
      </library>
```

Publication3.xml

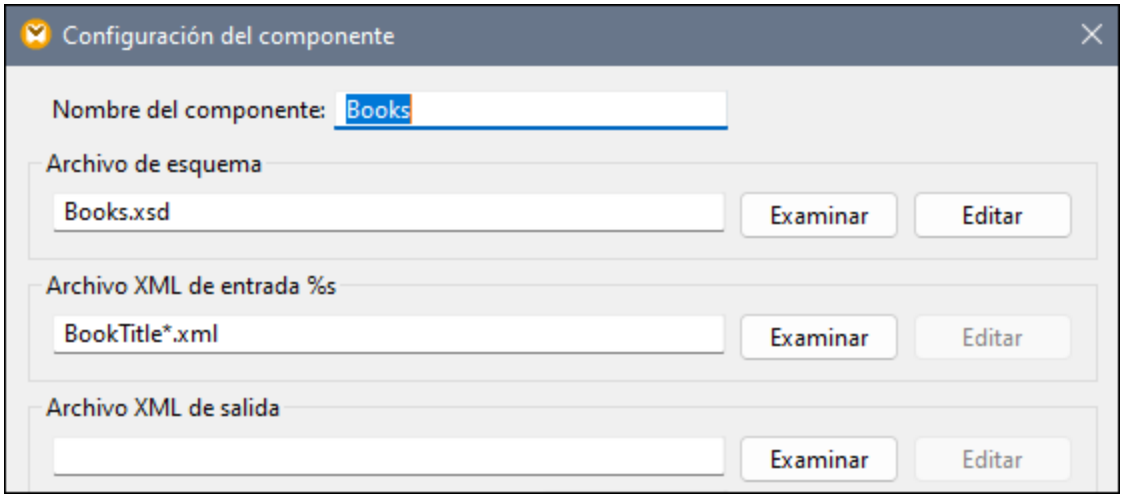
```
      <library>
         <publication>
            <id>3</id>
            <author>Herman Melville</author>
            <title>Moby Dick</title>
            <genre>Fiction</genre>
            <publish_year>1851</publish_year>
         </publication>
      </library>
```

Para llevar a cabo la transformación de datos siga los pasos que describimos en los apartados siguientes.

3.4.1 Configurar el componente de entrada

El primer paso consiste en modificar los ajustes del componente de origen. Antes de cambiar la configuración de los componentes, asegúrese de guardar la asignación con un nuevo nombre en la carpeta **Tutoriales básicos**.

Para indicar a MapForce que debe procesar varios archivos de instancia XML, haga doble clic en el encabezado del componente e introduzca **BookTitle*.xml** en el campo *Archivo XML de entrada* (*ver imagen siguiente*). El asterisco (*) del nombre del archivo indica a MapForce que use todos los archivos que tengan el prefijo `BookTitle` como archivos de entrada de la asignación. La ruta es relativa, por lo que MapForce buscará todos los archivos `BookTitle` en el mismo directorio en el que está el archivo de asignación, pero también puede usar una ruta absoluta si quiere.

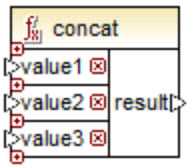


3.4.2 Configurar el componente de destino, parte 1

El paso siguiente es crear el nombre de archivo de los archivos de salida. Para ello vamos a usar la función **concat** 310 que combina todos los valores suministrados. Al unir estos valores se crea un nombre de archivo de salida (p. ej. **Publication1.xml**). Para generar los nombres de archivo con la función **concat** siga los pasos que describimos a continuación.

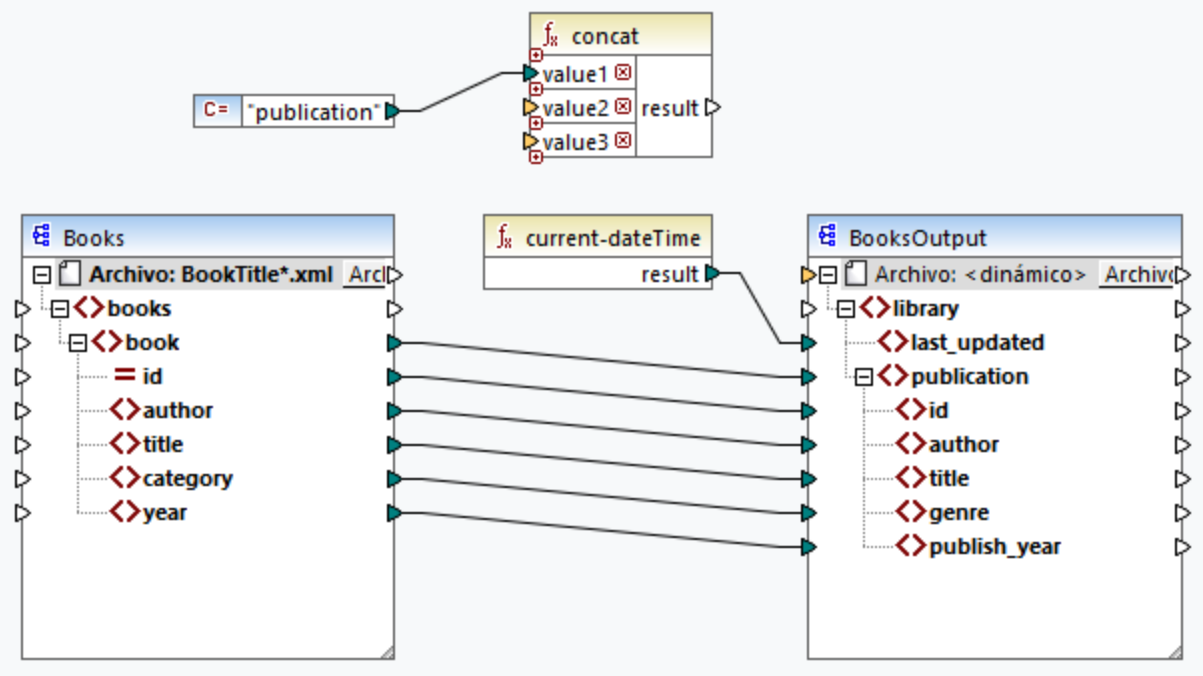
Paso 1: Agregar la función concat

Agregue 84 la función **concat** (*ver imagen siguiente*) al área de asignación. Por defecto esta función tiene dos parámetros cuando se añade a la asignación. En este ejemplo necesitamos tres parámetros. Haga clic en ⊡ (**Agregar parámetro**) dentro del componente de la función y añada un tercer parámetro. Para borrar parámetros use ⊠ (**Eliminar parámetro**).



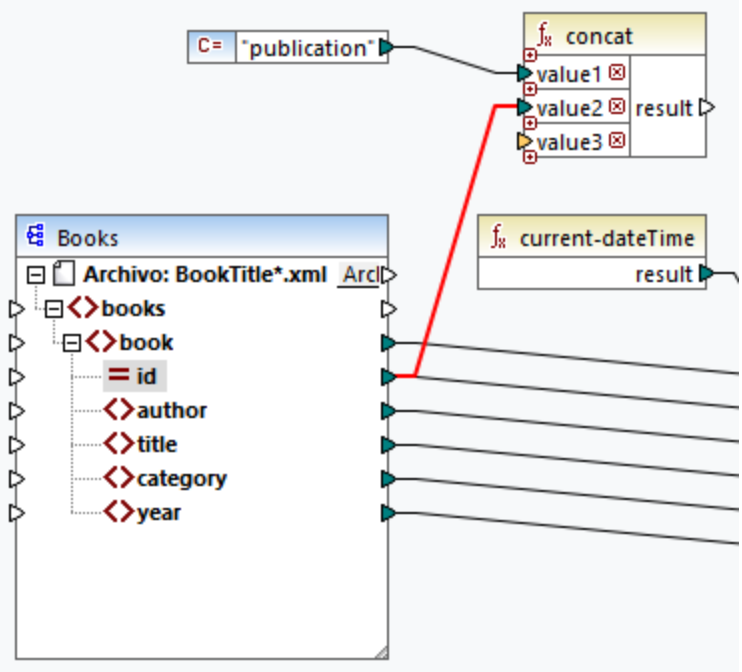
Paso 2: Insertar una constante

El siguiente paso consiste en agregar una constante. Cuando tenga que indicar un valor introduzca `publication` y deje la opción *String* como está. Conecte la constante con `value1` de la función **concat**, como se ve en la imagen siguiente:



Paso 3: Suministrar Ids

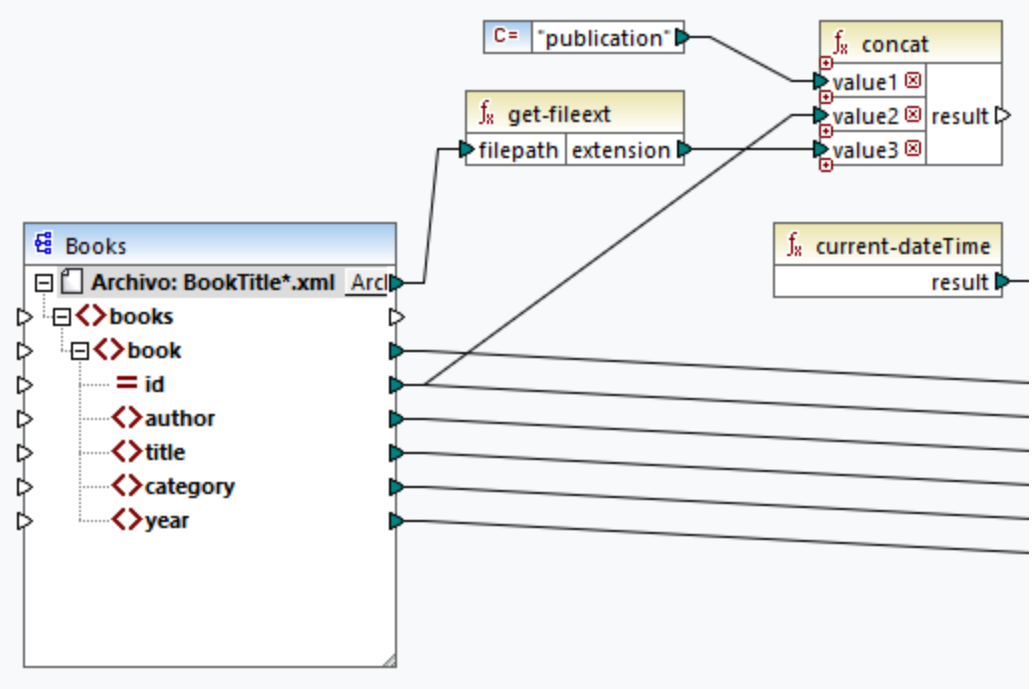
Conecte el atributo `id` del componente de origen con `value2` de la función **`concat`** (*ver imagen siguiente*). El atributo `id` del archivo XML de origen proporciona un valor de identificador único para cada uno de los archivos. Esta acción evita que los archivos se generen con el mismo nombre. La conexión se vuelve roja al hacer clic en ella.



Paso 4: Extraer la extensión de archivo

Agregue la función **`get-fileext`** 255 al área de asignación. Después cree una conexión desde del nodo superior del componente de origen (**`Archivo: BookTitle.xml`**) hasta el parámetro `filepath` de esta función (*ver imagen siguiente*).

Ahora conecte el parámetro `extension` de la función **`get-fileext`** a `value3` de la función **`concat`**. Así extrae solamente la parte de la extensión (en este caso, **`.xml`**) del nombre del archivo de origen y lo pasa al nombre del archivo de salida.

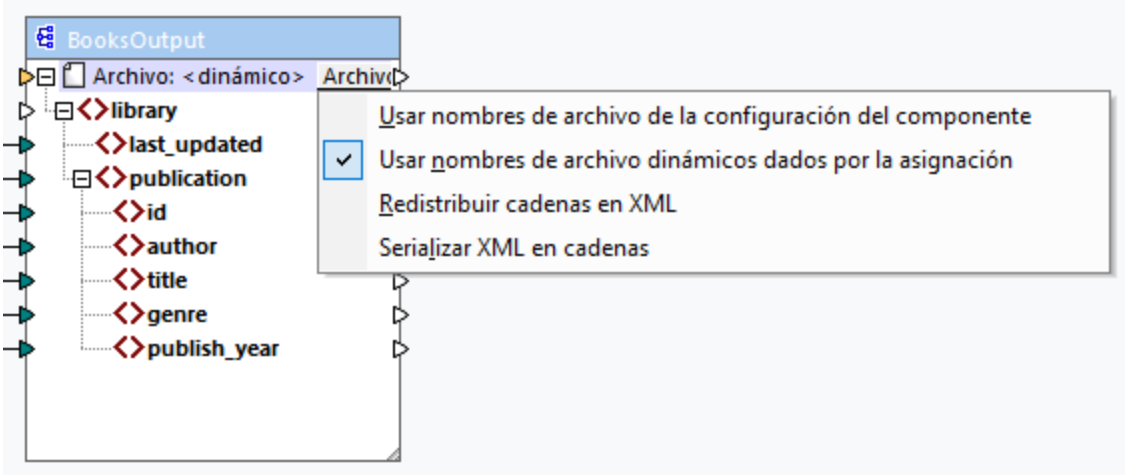


3.4.3 Configurar el componente de destino, parte 2

En la segunda parte de la configuración de salida, indicaremos a MapForce que genere archivos de salida de forma dinámica, lo que significa que los archivos de salida reciben sus respectivos nombres en base a los argumentos suministrados a la función **concat**. Para ello usaremos nombres de archivo dinámicos (*véase los puntos siguientes*). Para más información sobre nombres de archivo dinámicos consulte el apartado Procesar varios archivos de entrada y salida 402.

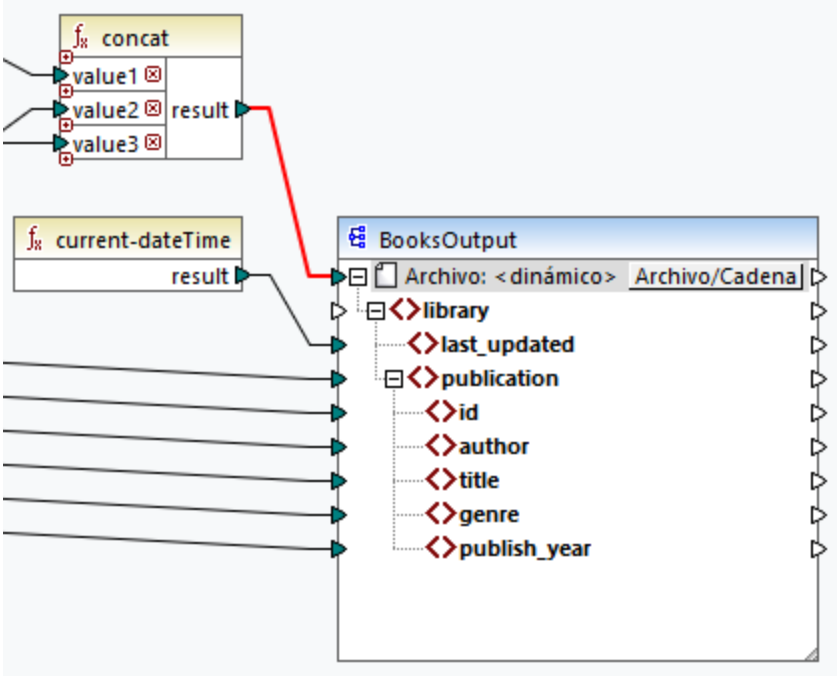
Paso 1: Configurar nombres de archivo dinámicos

Para que MapForce genere dinámicamente archivos, haga clic en el botón Archivo o Archivo/Cadena junto con el nodo `Archivo` del componente de destino y seleccione **Usar nombres de archivo dinámicos dados por la asignación** del menú contextual (*imagen siguiente*).



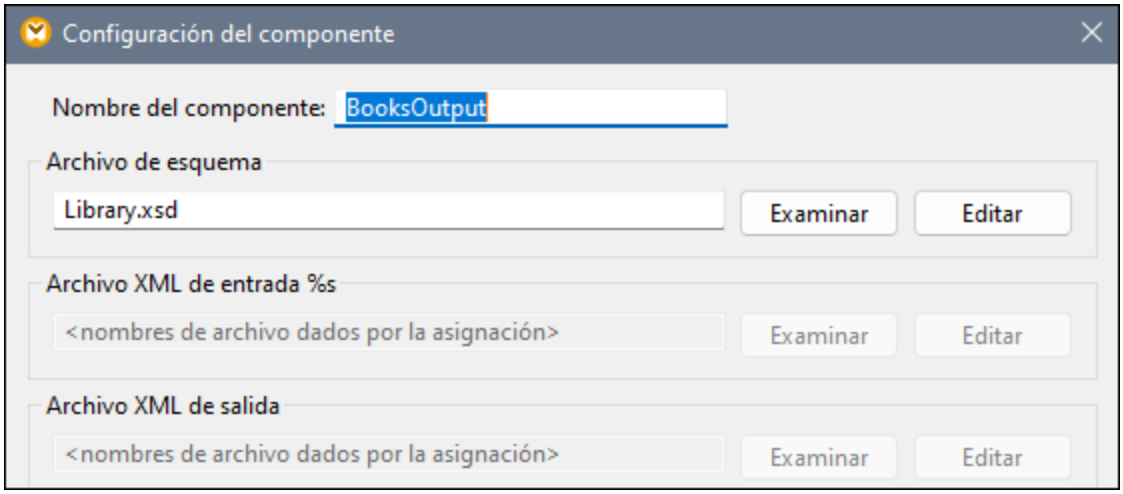
Paso 2: Conectar la función concat con un nodo dinámico

El paso siguiente es conectar el resultado de la función **`concat`** con el nodo `Archivo: <dynamic>` del componente de destino (*imagen siguiente*).



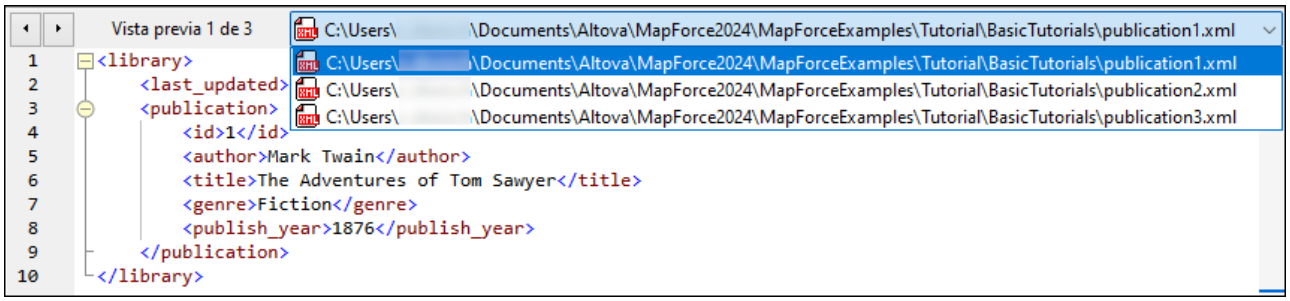
Paso 3: Comprobar configuración del componente

En la configuración del componente, notará que las casillas *Archivo XML de entrada* y *Archivo XML de salida* están deshabilitadas y aparece <*Nombre de archivo dado por la aplicación*> (*imagen siguiente*). Esto indica que los nombres de archivo de instancia se han obtenido de forma dinámica de la asignación. Por tanto, ya no es posible definirlos en la configuración del componente.



Paso 4: Generar archivos de salida

Ahora puede ejecutar la asignación y ver el resultado, así como los nombres de los archivos generados. Puede pasar de un archivo de salida a otro con los botones de izquierda y derecha que hay en la esquina superior izquierda del panel Resultados o haciendo clic en un archivo en la lista desplegable adyacente (*imagen siguiente*).



Hemos guardado el diseño de asignación de este tutorial como **`Tut4_MultipleToMultiple.mfd`**.

4 Componentes estructurales

En esta sección encontrará información sobre distintos formatos de datos que puede usar como orígenes y destinos de datos:

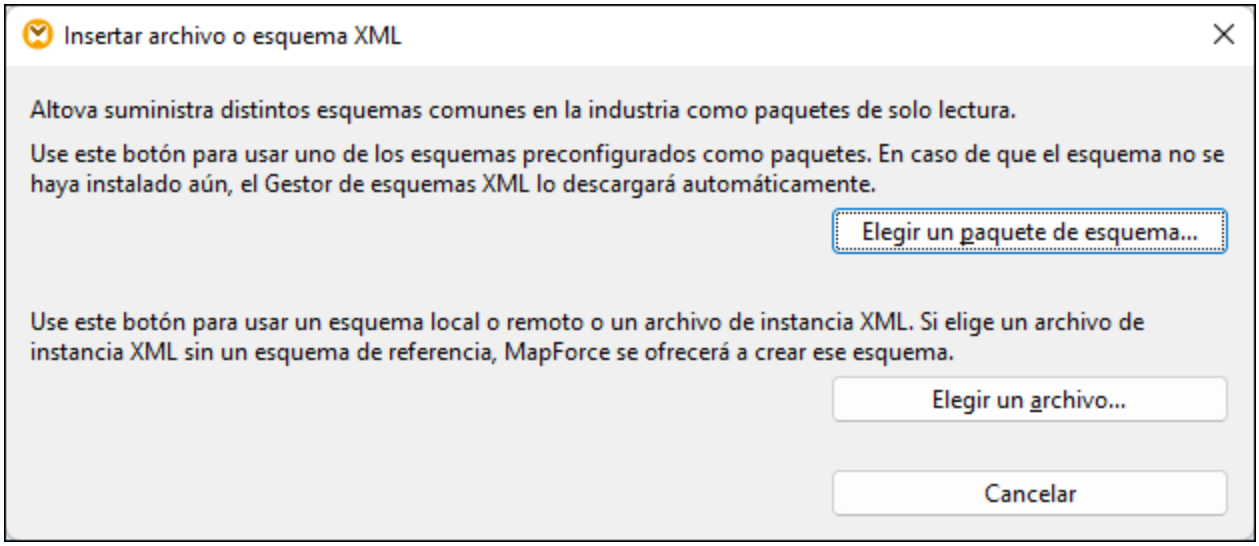
4.1 XML y esquemas XML

Sitio web de Altova: Asignación de datos XML

XML es un lenguaje de consultas para documentos de texto. XML Schema define la estructura y las restricciones de los documentos XML. En MapForce los archivos XML son componentes estructurales 32 que se pueden usar como orígenes y destinos de datos. Para más información sobre casos básicos de transformación de datos consulte los Tutoriales 78.

Insertar un archivo o esquema XML

Para insertar un archivo o esquema XML seleccione el comando de menú **Insertar | Archivo o esquema XML** o use el botón de la barra de herramientas. El cuadro de diálogo (*imagen siguiente*) le pedirá que elija entre un paquete de esquema estándar en el sector y un archivo de esquema o instancia remoto o local. Si elige el paquete de esquema deberá indicar un punto de entrada. Si el esquema que quiere usar todavía no está instalado, el Gestor de esquemas XML 130 lo descargará automáticamente.



Generar un esquema XML

Al agregar un archivo XML local o remoto sin referencia a un esquema, MapForce sugiere generar uno por usted. Si quiere que lo genere deberá seleccionar el directorio en el que quiere guardar el esquema generado.

Cuando MapForce genera un esquema a partir de un archivo XML, los tipos de datos para elementos y atributos deben inferirse del documento de instancia XML y puede que el resultado no sea exactamente lo que se esperaba. Por eso, recomendamos que terminada la operación compruebe si el esquema generado representa correctamente los datos de instancia.

Si existen elementos o atributos en más de un espacio de nombres, MapForce genera un esquema XML distinto por cada espacio de nombres (es decir, puede que se creen varios archivos en el disco).

DTD como estructura de un documento

A partir de la versión 2006 SP2, MapForce es compatible con el uso de documentos DTD preparados para espacios de nombres como componentes de origen y destino. Para ello, los URI de espacio de nombres se

extraen de las declaraciones de atributo `xmlns` de la DTD. Sin embargo, algunas DTD contienen declaraciones de atributo `xmlns` sin URI de espacio de nombres (p.ej. las DTD que utiliza StyleVision). Estas DTD se deben ampliar para poder utilizarse en MapForce. Concretamente los atributos `xmlns` deben definirse con el URI de espacio de nombres tal y como aparece en este ejemplo:

```
<!ATTLIST fo:root
   xmlns:fo CDATA #FIXED 'http://www.w3.org/1999/XSL/Format'
   ...
>
```

Nota sobre los valores de enumeración

Para los nodos cuyos tipos de datos tienen facetas de enumeración, puede crear una asignación de valores que tendrá todos los valores de enumeración ya rellenados. Así le resultará más fácil procesar y asignar valores de enumeración. Para más información consulte el apartado Asignación de valores 188.

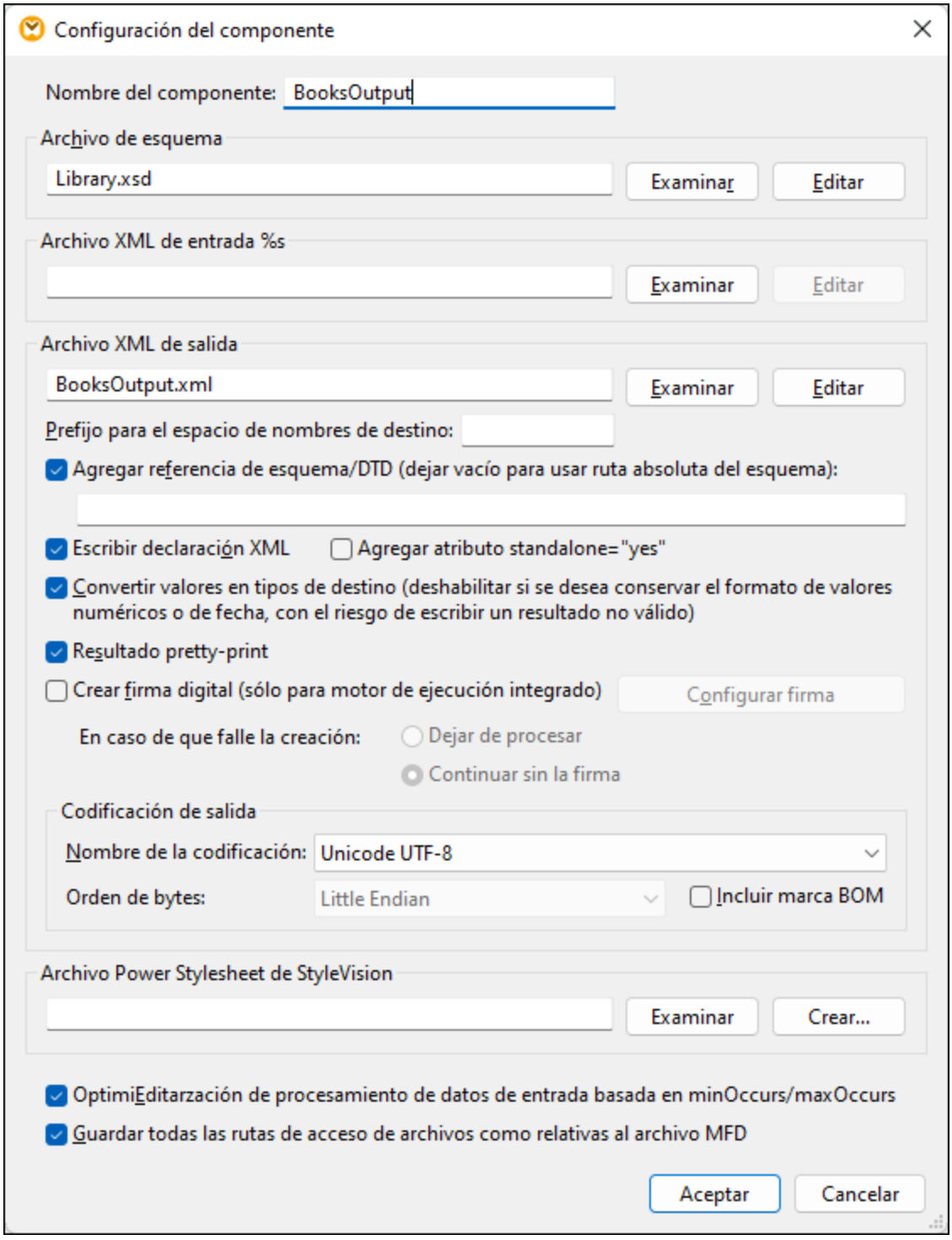
En esta sección

Estos son los apartados de esta sección:

4.1.1 Configuración de componentes XML

Una vez haya añadido un componente XML al área de la asignación puede configurarlo desde el cuadro de diálogo **Configuración del componente** (*imagen siguiente*). Puede abrir este cuadro de diálogo de una de las siguientes maneras:

- Haga doble clic en el encabezado del componente
- Haga clic con el botón derecho en el encabezado del componente y luego seleccione **Propiedades**
- Seleccione el componente en la asignación y haga clic en **Propiedades** en el menú **Componente**



A continuación puede ver las opciones que se pueden configurar en este cuadro de diálogo.

Configuración general

- Nombre del componente

El nombre del componente se genera automáticamente al crearlo, pero se puede volver a cambiar cuando quiera. El nombre del componente puede contener espacios y puntos, pero no barra, barra inversa, dos puntos, comillas dobles, ni espacios antes ni después del nombre.

Si cambia el nombre del componente, tenga en cuenta que:

- Si quiere implementar la asignación en FlowForce Server, el nombre del componente debe ser único.
- Recomendamos que use únicamente caracteres que se puedan usar también en la línea de comandos. Los caracteres especiales propios de cada país pueden estar codificados de forma diferente en Windows y la línea de comandos.

Archivo del esquema

Indica el nombre o la ruta de acceso del archivo de esquema XML que MapForce usa para validar y asignar los datos. Para cambiar el archivo del esquema haga clic en **Examinar** y seleccione un archivo nuevo. Para editar el archivo en Altova XMLSpy haga clic en **Editar**.

Archivo XML de entrada

Indica el archivo de instancia XML del que MapForce debe leer los datos. Este campo es relevante para los componentes de origen y se rellena cuando se crea el componente y se le asigna un archivo de instancia XML. En un componente de origen el nombre del archivo de instancia también se usa para detectar el elemento raíz de la instancia XML y el esquema al que se hace referencia, así como para validar los datos con el esquema seleccionado. Para cambiar el archivo del esquema haga clic en **Examinar** y seleccione un archivo nuevo. Para editar el archivo en Altova XMLSpy haga clic en **Editar**.

Archivo XML de salida

Indica el archivo de instancia XML en el que MapForce escribirá los datos. Este campo es significativo para un componente de destino. Para cambiar el archivo del esquema haga clic en **Examinar** y seleccione un archivo nuevo. Para editar el archivo en Altova XMLSpy haga clic en **Editar**.

Prefijo para espacios de nombres de destino

Permite introducir un prefijo para el espacio de nombres de destino. Antes de asignar el prefijo debe asegurarse de que el espacio de nombres de destino está definido en el esquema de destino.

Agregar referencia de esquema/DTD

Agrega la ruta del archivo de esquema XML al que se hace referencia al elemento raíz de los datos XML de salida. La ruta del esquema que se introduce en este campo se escribe en los archivos de instancia de destino que se generan en el atributo `xsi:schemaLocation` o en la declaración `DOCTYPE` si se está usando una DTD.

Ediciones MapForce Professional y Enterprise: Si genera código en XQuery o C++ no se permite agregar también la referencia de la DTD.

Si introduce una ruta en este campo puede definir dónde se encuentra el archivo de esquema al que hace referencia el archivo de instancia XML. Esto permite poder validar el archivo de instancia de salida en la ubicación de destino de la asignación cuando esta se ejecute. En el campo Agregar referencia de esquema/DTD puede introducir una dirección **`http://`** o una ruta de acceso absoluta o relativa

Si desactiva esta opción puede desconectar la instancia XML de los esquemas XML o documentos DTD a los que se hace referencia. Esto puede ser útil, por ejemplo, si quiere enviar los datos XML de salida a alguien que no tiene acceso al esquema XML subyacente.

- Escribir una declaración XML

Esta opción está habilitada por defecto, lo que significa que la declaración XML se escribe en la salida. En la tabla siguiente puede ver la compatibilidad de esta funcionalidad con los lenguajes de destino de MapForce y sus motores de ejecución.

Lenguaje de destino/motor de ejecución	**Si el resultado es un archivo**	**Si el resultado es una cadena de texto**
Integrado (*ediciones Professional y Enterprise*)	Sí	Sí
MapForce Server (*ediciones Professional y Enterprise*)	Sí	Sí
XQuery (*ediciones Professional y Enterprise*), XSLT	Sí	No
Generador de código (C++, C#, Java) (*ediciones Professional y Enterprise*)	Sí	Sí

- Agregar independiente="yes"

Si selecciona esta opción se inserta la declaración `standalone="yes"` en la declaración XML del archivo XML de destino. Para saber más consulte la *declaración de documento independiente*.

Tenga en cuenta que:

- Cuando se selecciona la opción *independiente="yes"*, la generación de datos de salida es compatible con el lenguaje Built-In XSLT 1-3 y con la generación de código (C#, Java, C++ MSXML, C+ Xerces). El lenguaje de transformación Built-In[17] y el código generado están disponibles en las ediciones Professional y Enterprise.
- No hay compatibilidad con XML incrustado en campos de bases de datos y solicitudes de servicios web (*ediciones Professional y Enterprise*).

- Convertir valores en tipos de destino

Permite (i) definir si los tipos del esquema XML de destino deben utilizarse durante la asignación o (ii) si todos los datos asignados al componente de destino deben considerarse valores de cadena. Por defecto, esta opción está habilitada. Si desactiva esta opción podrá conservar el formato preciso de los valores. Por ejemplo, si se desea satisfacer una faceta de patrón de un esquema que requiere un número concreto de dígitos decimales en un valor numérico. Puede usar funciones de asignación de datos para dar el formato de cadena necesario al número y después asignar esta cadena al destino.

Recuerde que si deshabilita esta opción, también se deshabilita la detección de valores no válidos (p. ej. si se escriben letras en campos numéricos).

- Resultado pretty-print

Ajusta el formato del documento XML de salida para que tenga un aspecto estructurado. Cada nodo secundario se presenta alejado de su elemento primario por cuatro caracteres de espacio.

- Crear firma digital (*Enterprise Edition*)

  Permite agregar una firma digital al archivo XML de instancia de salida. Esto sólo es compatible con el lenguaje de transformación integrado.

Codificación de salida

Permite especificar las siguientes opciones del archivo de instancia de salida:

- Nombre de la codificación
- Orden de bytes
- Si se incluye o no la marca BOM

La codificación de cualquier componente nuevo está definida por defecto en la opción *Codificación predeterminada para componentes nuevos*. Puede acceder a esta opción desde **Herramientas | Opciones** (sección *Generales*).

Si la asignación genera XSLT 1.0/2.0, marcar la casilla *Orden de bytes* no tiene ningún efecto, ya que estos lenguajes no son compatibles con el orden de bytes.

Archivo hoja de estilos StyleVision Power

Esta opción permite seleccionar o crear un archivo de hoja de estilos de Altova StyleVision Estos archivos permiten presentar los datos de salida de un archivo XML de instancia en varios formatos, como HTML, RTF, etc. Consulte también el apartado Usar rutas de acceso relativas en un componente 42.

Otras opciones

- Optimización de procesamiento de datos de entrada basada en minOccurs/maxOccurs

  Esta opción permite un manejo especial de las secuencias de las que se sabe que contienen exactamente un elemento, como atributos necesarios o elementos secundarios con `minOccurs` y `maxOccurs="1"`. En este caso se extrae el primer elemento de la secuencia y se procesa directamente como un valor atómico (y no como una secuencia)

  Si los datos de entrada son **no válidos** para el esquema, la asignación genera un mensaje de error. Para permitir que se procesen datos de entrada **no válidos**, desmarque esta casilla.

- Guardar todas las rutas de acceso de archivos como relativas al archivo MFD

  Cuando esta opción está habilitada, MapForce guarda las rutas de acceso al archivo que aparecen en el cuadro de diálogo **Configuración del componente** relativas a la ubicación del archivo de diseño de MapForce (**`.mfd`**). Consulte también Rutas de acceso relativas en un componente 42.

4.1.2 Tipos derivados

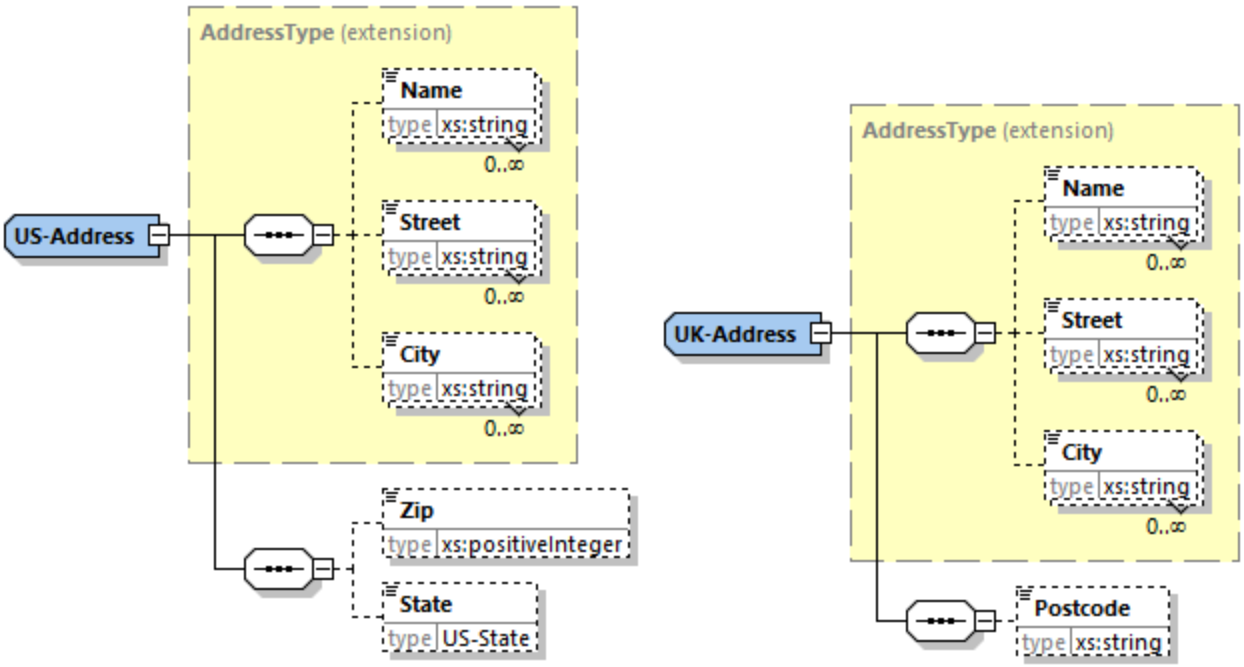
MapForce permite realizar asignaciones de datos entre tipos derivados de un tipo complejo. Los tipos derivados son tipos complejos de un esquema XML que usan el atributo `xsi:type` para identificar los tipos derivados especificados (p.e., `<Address xsi:type="UK-Address">`). Puede ver las definiciones de estos tipos en la especificación de esquema XML de W3C (sección 2.5.2).

Caso posible

En esta subsección explicamos un posible caso de uso de un tipo derivado. Por ejemplo, imagine que hay una empresa con dos filiales: una en Reino Unido y la otra en Estados Unidos. Lo que queremos es tener dos listas (`UKCustomers` y `USCustomers`), cada una de las cuales debe incluir la información sobre la dirección de la filial correspondiente y todos los clientes asociados a ella.

Definición de tipos derivados

En las imágenes siguientes puede ver la definición de los tipos derivados `US-Address` y `UK-Address` (*en la vista Esquema de XMLSpy*). El tipo base (o tipo complejo originario) es `AddressType` e incluye los elementos `Name`, `Street` y `City`. En el elemento `US-Address` se añadieron también dos elementos más para crear los tipos derivados `US-Address`: `Zip` y `State`, mientras que el elemento `UK-Address` incluye el tipo base y el elemento `Postcode`. En este ejemplo sólo vamos a asignar al archivo de destino el elemento `UK-Address`.

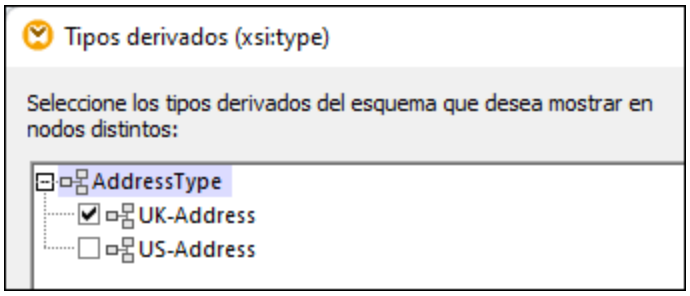


Tipos derivados en una asignación

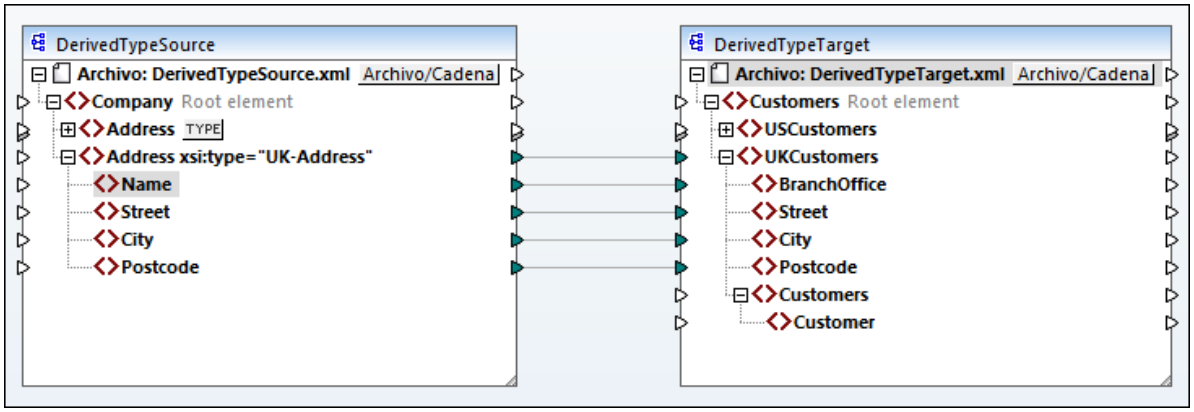
A continuación explicamos cómo realizar asignaciones de datos entre tipos derivados de esquema XML. Lo que buscamos es asignar información sobre la oficina de Reino Unido al elemento `UKCustomers`. Puede encontrar los archivos de ejemplo en la carpeta **`Tutorial`**.

1. En el menú **Insertar** haga clic en el comando **Archivo o esquema XML** y abra **`DerivedTypeSource.xml`**. Este archivo XML se basa en **`DerivedTypeSource.xsd`**..

2. Inserte el archivo de destino **`DerivedTypeTarget.xsd`**. Observe que el esquema de destino no necesita incluir el atributo `xsi:type`.
3. Ahora haga clic en el botón TYPE situado junto al elemento `Address`. Este botón indica que para este elemento existen tipos derivados en el esquema.
4. EL cuadro de diálogo Tipos derivados (*imagen siguiente*) permite seleccionar los tipos derivados disponibles para ese elemento en concreto. En esta asignación de ejemplo solamente queremos asignar el elemento `UK-Address`.



5. Al marcar la casilla del tipo derivado `UK-Address` verá que en el componente aparece un elemento nuevo llamado `Address xsi:type="UK-Address"`.
6. Conecte los nodos como se ver en la imagen siguiente.



Resultado
Al hacer clic en el panel Resultados verá este código:

```
<UKCustomers>
   <BranchOffice>Sleuth Corp. UK</BranchOffice>
   <Street>222 Baker St</Street>
   <City>London</City>
   <Postcode>NW1 6XE</Postcode>
</UKCustomers>
```

Esta asignación de ejemplo está guardada como **`Tutorial\DerivedType.mfd`**. También puede agregar otro archivo XML de origen que incluya información sobre los clientes de Reuno Unido y asignar esos datos al nodo `Customers` del componente de destino. De esta manera el elemento `UKCustomers` incluye información sobre la dirección de Reino Unido y todos los clientes asociados a esa filial.

4.1.3 Valores NULL

En esta sección explicamos cómo se ocupa MapForce de los valores NULL en los componente de origen y de destino. Para poder usar el atributo `xsi:nil="true"` en un archivo XML debe especificar el atributo `nillable="true»` en los elementos relevantes del archivo de esquema correspondiente. Para saber más sobre los atributos `nillable` y `xsi:nil` consulte la especificación del W3C. Observe que el atributo `xsi:nil` no es visible en la estructura de los componentes del panel **Asignación**.

A continuación explicamos algunos casos de uso de los valores NULL en asignaciones.

Valores NULL en componentes XML

En esta subsección vamos a ver algunos ejemplos de casos de uso de elementos de asignación que tienen un atributo `xsi:nil="true"`.

Sólo el elemento de origen tiene xsi:nil="true"/Tanto el elemento de origen como el de destino tienen xsi:nil="true"
En este caso las condiciones son:

- La conexión está basada en destino[50].
- El elemento de origen es un atributo `xsi:nil="true"`. El elemento de destino correspondiente no tiene este atributo.
- También es posible que los dos elementos, el de origen y el de destino, tengan atributos `xsi:nil="true"`.
- Los atributos `nillable="true"` se deben definir en los esquemas de origen y de destino,
- Los elementos de origen y de destino son de tipo simple.

En este caso, el elemento de destino tendrá el atributo `xsi:nil="true"` en el archivo de salida, como se ve en el archivo de resultados de más abajo (*resaltado en amarillo*).

```
<book id="7">
  <author>Edgar Allan Poe</author>
  <title>The Murders in the Rue Morgue</title>
  <category xsi:nil="true"/>
  <year>1841</year>
  <OrderID id="213"/>
</book>
```

Nota: Si el atributo `nillable="true"` no está definido en el esquema de destino, el elemento de destino correspondiente estará vacío en el resultado.

Sólo el elemento de destino tiene xsi:nil="true»
En este caso las condiciones son:

- La conexión está basada en destino[50].
- El elemento de origen no tiene un atributo `xsi:nil="true"`.
- El elemento de destino correspondiente tiene un atributo `xsi:nil="true"`.
- Los elementos de origen y de destino pueden ser de tipo simple o complejo.

En este caso el elemento de origen sobrescribirá el elemento de destino que contiene el atributo `xsi:nil="true"`. En el ejemplo siguiente puede ver un archivo de resultados de ejemplo. El elemento `<genre>`

incluye el atributo `xsi:nil="true"` en el elemento de destino. Sin embargo, este elemento se sobrescribe en tiempo de ejecución, por lo que el elemento `<genre>` (*resaltado en amarillo*) tiene `Fiction` en el resultado.

```
<publication>
  <id>1</id>
  <author>Mark Twain</author>
  <title>The Adventures of Tom Sawyer</title>
  <genre>Fiction</genre>
  <year>1876</year>
  <OrderID id="124"/>
</publication>
```

El elemento de tipo complejo de origen/los dos elementos de tipo complejo tienen xsi:nil="true"
En este caso las condiciones son:

- La conexión está basada en destino[50].
- El elemento de origen es de tipo complejo. En este ejemplo el elemento de origen tiene un atributo `id="213"` y un atributo `xsi:nil="true»`. El elemento de destino correspondiente también es de tipo complejo y tiene un atributo `id="124"`, pero no un atributo `xsi:nil="true»`.
- También puede ser que los dos elementos, el de origen y el de destino, que son de tipo complejo, tengan atributos `xsi:nil="true»`.

En este caso el elemento de origen sobrescribirá el elemento de destino (*resaltado en amarillo*). Sin embargo, el atributo `xsi:nil="true"` no se escribe automáticamente en el archivo de salida. Para ver el atributo `xsi:nil="true"` en el elemento de destio del archivo de resultados use una conexión de copia total[55].

```
<book id="7">
  <author>Edgar Allan Poe</author>
  <title>The Murders in the Rue Morgue</title>
  <year>1841</year>
  <OrderID id="213"/>
</book>
```

Funciones útiles

Estas funciones sirven para comprobar, reemplazar y asignar valores NULL:

- **`is-xsi-nil`**[273]: Ayuda a comprobar de forma explícita si un elemento de origen tiene el atributo `xsi:nil` definido como `true`.
- **`substitute-missing`**[307]: Sustituye un valor NULL en el elemento de origen con algo específico.
- **`set-xsi-nil`**[276]: Asigna el elemento `xsi:nil="true"` a un elemento de destino. Esto funciona con elementos de destino de tipo simple y complejo.
- **`substitute-missing-with-xsi-nil`**[277]: Si hay contenido, este se escribe en el elemento de destino; si falta algún valor, esta función produce un elemento de destino con el atributo `xsi:nil="true»` en el resultado.
- Si conecta la función **`exists`**[281] a un elemento de origen con un valor NULL devuelve `true` aunque el elemento no tenga contenido.

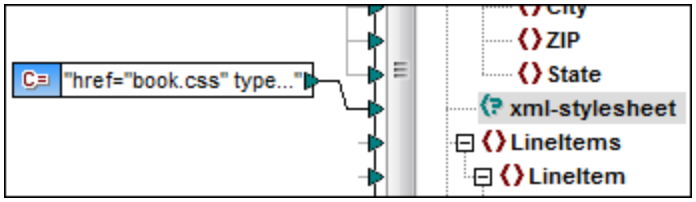
Observe que las funciones que generan `xsi:nil` no pueden pasar a través de funciones o componentes que sólo operan en valores (como la función **`if-else`**).

4.1.4 Comentarios e instrucciones de procesamiento

En los componentes XML de destino se pueden insertar comentarios e instrucciones de procesamiento (consulte la especificación de W3C). Estas últimas sirven para pasar información a las aplicaciones que continúan con el procesamiento de los documentos XML. Recuerde que no se pueden definir comentarios ni instrucciones de procesamiento en nodos que formen parte de un grupo de asignación de copia total [55].

Para insertar una instrucción de procesamiento
Para insertar una instrucción de procesamiento siga estos pasos:

1. Haga clic con el botón derecho en un elemento del componente de destino y elija **Comentario/instrucción de procesamiento** en el menú contextual y después elija **Agregar delante una instrucción de procesamiento** o bien **Agregar detrás una instrucción de procesamiento**. Introduzca el nombre (de destino) de la instrucción de procesamiento y haga clic en **Aceptar** para confirmar. En este ejemplo hemos insertado la instrucción `xml-stylesheet` e la estructura del componente de destino, tras el elemento `State`.



2. Ahora puede usar un componente de constante para aportar el valor del atributo de la instrucción de procesamiento (*imagen anterior*).

Nota: Puede añadir varias instrucciones de procesamiento delante o detrás de cualquier elemento del componente de destino.

Nota: Sólo se puede añadir un comentario antes y después de un nodo de destino. Para crear varios comentarios use la función de duplicar componentes de entrada [40].

Para eliminar un comentario o una instrucción de procesamiento
Haga clic con el botón derecho en el correspondiente nodo, elija **Comentario/instrucción de procesamiento** en el menú contextual y después elija **Eliminar**.

4.1.5 Secciones CDATA

Las secciones CDATA sirven para añadir caracteres de escape a bloques de texto que contienen caracteres que se podrían interpretar como marcado. Para más información consulte la especificación de W3C. Los nodos de destino pueden escribir los datos de entrada que reciben como secciones CDATA. Los componentes de destino pueden ser: datos XML, datos XML incrustados en campos de BD o elementos secundarios XML de dimensiones con tipo de un componente XBRL de destino. También se pueden definir secciones CDATA en nodos duplicados y en nodos `xsi:type`.

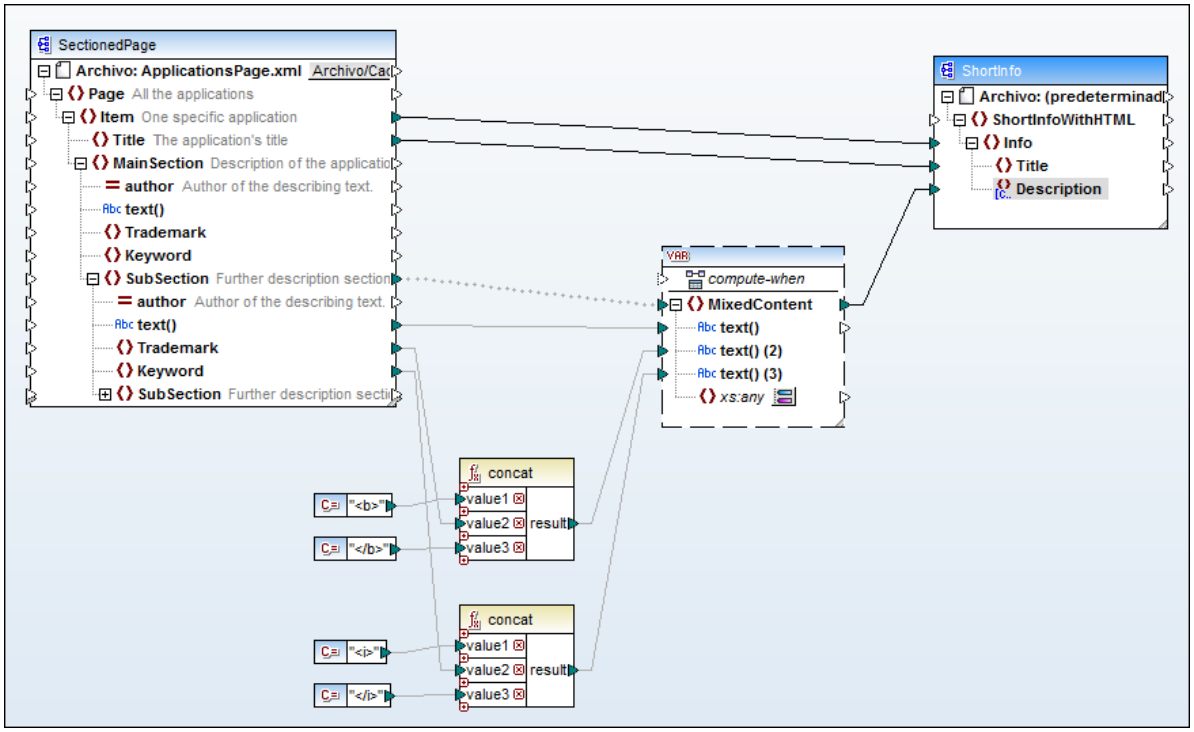
Para crear una sección CDATA haga clic con el botón derecho en el nodo de destino que desea definir como sección CDATA y seleccione **Escribir contenido como sección CDATA**. Aparece entonces una advertencia señalando que los datos de entrada no deben contener el delimitador de cierre de sección CDATA `]]>`. Haga

clic en **Aceptar** para cerrar el aviso. El icono **[C..** que aparece debajo de la etiqueta del elemento indica que este nodo está definido como sección CDATA.

Ejemplo

El diseño de asignación **MapForceExamples\HTMLinCDATA.mfd** muestra lo útil que pueden ser las secciones CDATA (*imagen siguiente*). En este ejemplo:

- El elemento `SubSection` se definió como nodo basado en el origen (de contenido mixto).
- Al contenido del elemento de origen `Trademark` se añaden las etiquetas de apertura y cierre `<b></b>`.
- Al contenido del elemento de origen `Keyword` se añaden las etiquetas de apertura y cierre `<i></i>`
- Los datos resultantes se pasan a los nodos `text()` duplicados en el orden en que aparecen en el documento de origen porque el conector del elemento `SubSection` se definió como nodo basado en el origen (de contenido mixto).
- El resultado del nodo `MixedContent` se pasa después al nodo `Description` del componente de destino `ShortInfo`, que se definió como sección CDATA.



Resultado

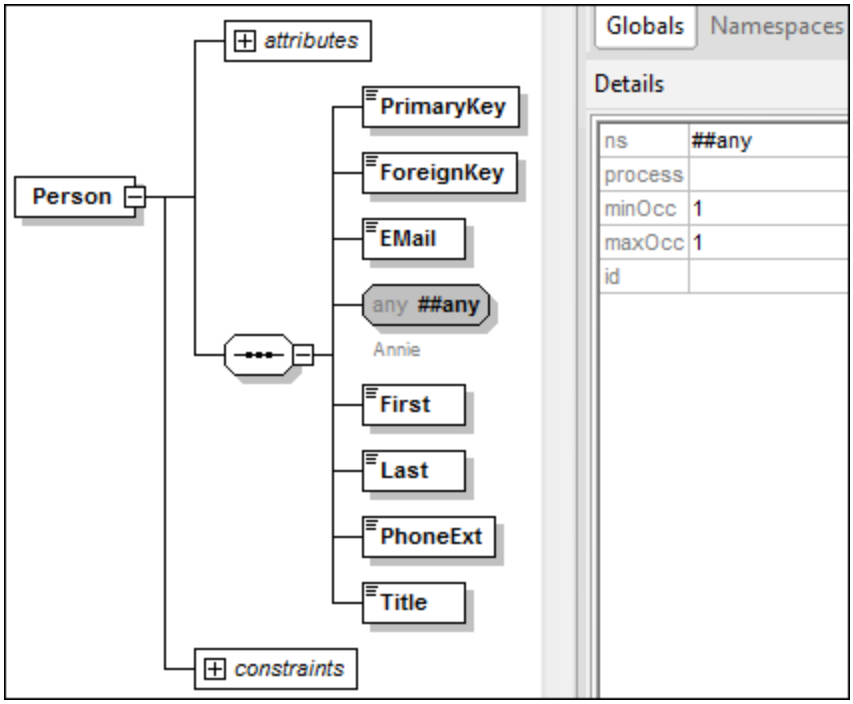
Si abrimos el panel *Resultados* podremos ver la sección CDATA que contiene el texto con marcado.

```
7     <Info>
8       <Title>MapForce</Title>
9       <Description><![CDATA[Altova <b>MapForce</b> 2014 Enterprise Edition is the premier <i>XML</i>
      / <i>database</i> / <i>flat file</i> / <i>EDI</i> data mapping tool that auto-generates mapping code in
      <i>XSLT</i> 1.0/2.0, <i>XQuery</i>, <i>Java</i>, <i>C++</i> and <i>C#</i>. It is the definitive tool for
      data integration and information leverage.]]></Description>
10    </Info>
```

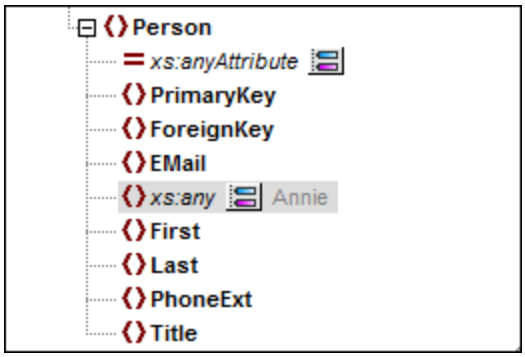
4.1.6 Comodines: xs:any / xs:anyAttribute

En este apartado explicamos cómo trabajar con comodines en asignaciones de datos. Los comodines `xs:any` (y `xs:anyAttribute`) permiten usar cualquier elemento o atributo de los esquemas. más información consulte la especificación de W3C.

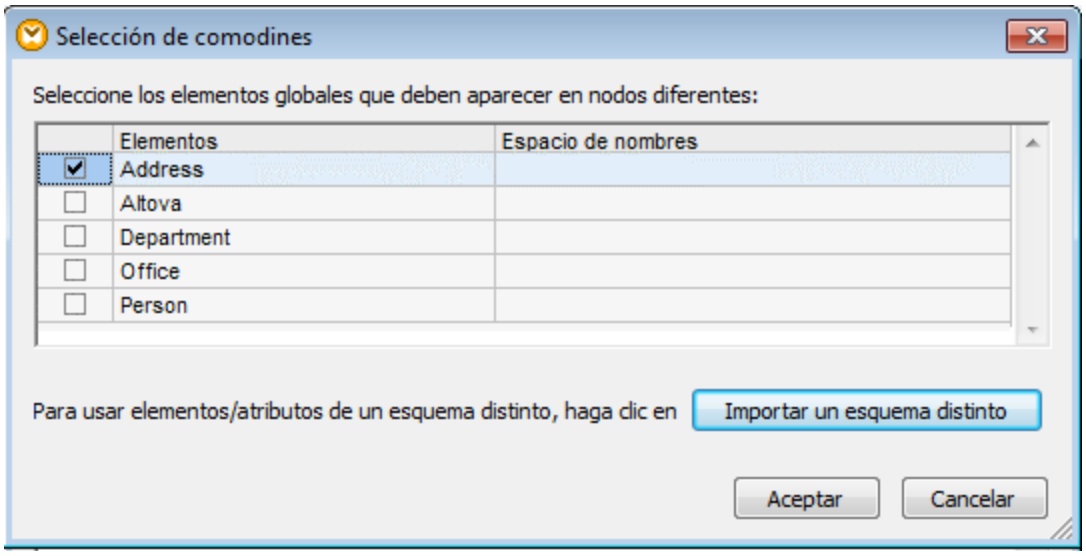
Comodines en la definición del esquema
En la imagen siguiente puede ver el elemento `xs:any` en la vista Esquema de Altova XMLSpy. Este elemento se ha definido como elemento secundario del elemento `Person`.



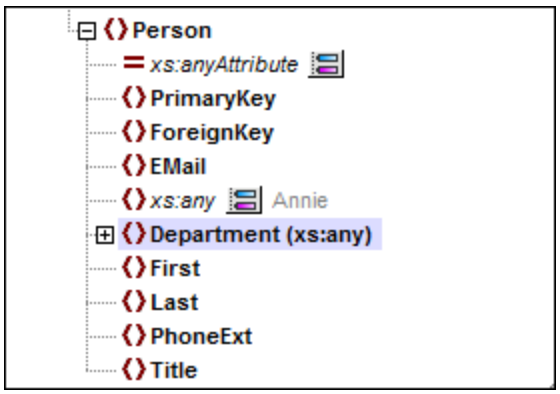
Comodines en MapForce
Cuando se define un comodín para un elemento y/o atributo, el nodo comodín aparece con el icono (**Cambiar selección**) junto a él (*imagen siguiente*).



Selección de comodines
Cuando se hace clic en el botón **Cambiar selección**, aparece el cuadro de diálogo "Selección de comodines" (*imagen siguiente*). En este cuadro de diálogo se enumeran los elementos y atributos globales declarados en el esquema actual.



En este ejemplo hemos seleccionado `Department`. Observe que los elementos y atributos comodines se insertan tras el nodo al que acompaña el icono . Ahora el componente tiene este aspecto:



Ahora puede crear asignaciones entre estos nodos y cualquier otro elemento. En un componente puede reconocer los elementos y atributos comodines por el texto `(xs:any) y (xs:anyAttribute)` que aparece anexado (*imagen anterior*).

Eliminar comodines

Para eliminar un elemento comodín haga clic en el botón **Cambiar selección** y desactive su casilla en el cuadro de diálogo "Selección de comodines".

Agregar elementos de un esquema distinto como comodines

El cuadro de diálogo "Selección de comodines" (*véase más arriba*) permite usar como comodines los elementos de un esquema que no sea el asignado. Para poder ver estos elementos en el componente haga clic en el botón **Importar un esquema distinto**, donde podrá elegir entre (i) importar un archivo de esquema o (ii) generar un esquema contenedor (*véase más abajo*).

Importar un esquema

La opción **Importar esquema** importa el esquema externo en el esquema actual que está asignado al componente. Tenga en cuenta que esta opción sobrescribe el esquema actual del componente en el disco. Si el esquema actual es un esquema remoto que se abrió desde una dirección URL (véase Agregar componentes desde una URL 36) y no desde el disco, entonces no se podrá modificar. En este caso debe utilizarse la opción **Generar esquema contenedor**.

Generar un esquema contenedor

La opción **Generar esquema contenedor** crea un archivo de esquema nuevo que se denomina esquema *contenedor*. La ventaja de usar esta opción reside en que el esquema actual del componente no se modifica. Por el contrario, se creará un esquema nuevo (es decir, un esquema contenedor) que incluirá tanto el esquema actual como el esquema que se desea importar. Cuando haga clic en esta opción, MapForce preguntará dónde se debe guardar el esquema contenedor. El nombre predeterminado del esquema contenedor será **`somefile-wrapper.xsd`**.

Una vez guardado, el esquema contenedor se asigna automáticamente y por defecto al componente y aparece un aviso que le pregunta si quiere cambiar la ubicación del esquema para poder hacer referencia al esquema principal anterior. Haga clic en **Sí** para volver al esquema anterior y en **No** para que el esquema contenedor recién creado siga asignado al componente.

Comodines y nombres de nodo dinámicos

En algunos casos los elementos o atributos que aparecen en la instancia son demasiados. Por ejemplo, en esta instancia:

```
<?xml version="1.0" encoding="UTF-8"?>
<message>
   <line1>1</line1>
   <line2>2</line2>
   <line3>3</line3>
   ...............
   <line999></line999>
</message>
```

Para casos como este, en lugar de comodines, se recomienda un acceso dinámico a los nombres de nodo (*véase Asignar nombres de nodos 385*).

4.1.7 Espacios de nombres personalizados

Cuando una asignación produce un resultado XML, MapForce deriva automáticamente el espacio de nombres (o conjunto de espacios de nombres) de cada elemento y atributo a partir del esquema asociado al componente de destino 30. Este es el comportamiento predeterminado de MapForce y el más apropiado en

las asignaciones de datos que implican la generación de resultados XML. Sin embargo, en otros casos puede ser preferible tener un mayor control sobre el espacio de nombres de los elementos en el código XML resultante. Por ejemplo, en algunos casos puede ser necesario declarar a mano el espacio de nombres de un elemento desde la asignación directamente.

La declaración de espacios de nombres personalizados (y el comando **Agregar espacio de nombres**) solamente es relevante para componentes de destino XML y solamente para elementos. El comando **Agregar espacio de nombres** no está disponible en el caso de atributos ni nodos comodín. Tampoco está disponible en caso de nodos que reciben datos por medio de conexiones de copia total 55.

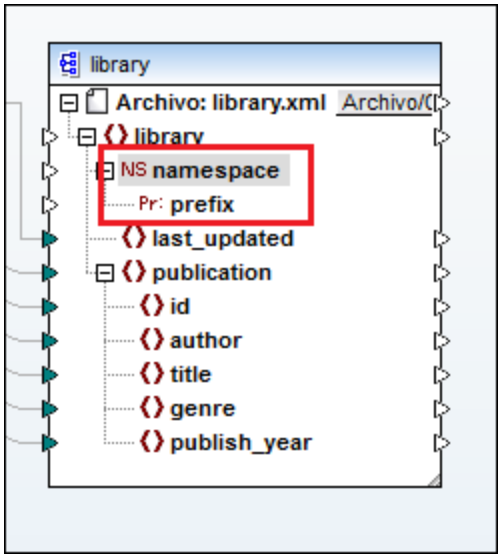
Para comprender mejor este funcionamiento siga las instrucciones de las subsecciones que siguen.

Declarar un espacio de nombres manualmente

Para este ejemplo necesita la asignación **`BasicTutorials\Tut1-SchemaToSchema.mfd`**.

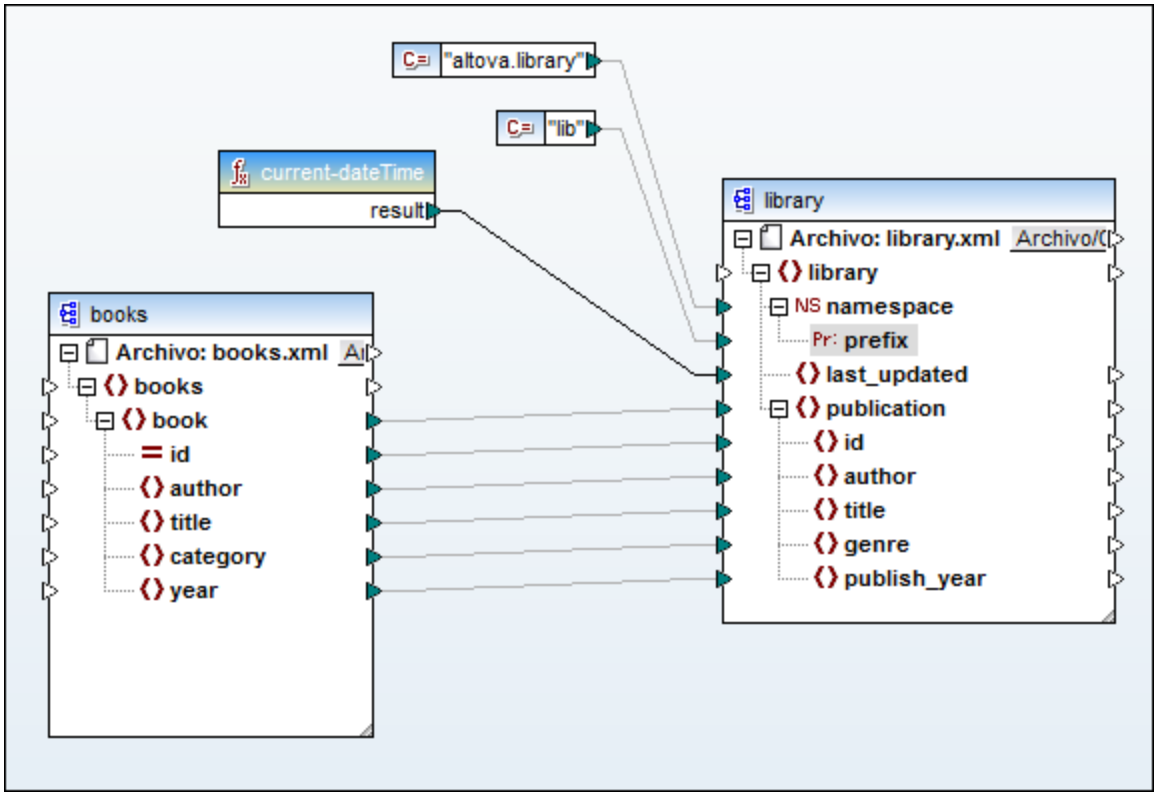
Agregar un espacio de nombres

Abra la asignación, haga clic con el botón derecho en el nodo `library` del componente `BooksOutput` y seleccione **Agregar espacio de nombres** en el menú contextual. Observe que ahora aparecen dos nodos nuevos bajo el nodo `library`: el nodo `namespace` y el nodo `prefix` (*imagen siguiente*).



Indicar los valores del espacio de nombres

Ahora puede crear asignaciones entre estos nodos y valores de cadena. Por ejemplo, en la imagen siguiente puede ver que se definieron dos constantes que aportan el espacio de nombres `altova.library` y el prefijo `lib` (*imagen siguiente*).



Nota: debe asignar los conectores de entrada `namespace` y `prefix` aunque les asigne valores vacíos.

Resultado
En el resultado podrá ver que el atributo `xmlns:<prefix>="<namespace>"` se añadió al elemento y que `<prefix>` y `<namespace>` son valores procedentes de la asignación. El resultado será (*fíjese en la parte resaltada*):

```
<?xml version="1.0" encoding="UTF-8"?>
<library xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:lib="altova.library" xsi:noNamespaceSchemaLocation="Library.xsd">
...
```

También puede declarar varios espacios de nombres para un mismo elemento. Para ello vuelva a hacer clic con el botón derecho en el nodo y seleccione **Agregar espacio de nombres** en el menú contextual. Bajo el nodo aparecerán dos nodos más para el espacio de nombres y para el prefijo. Ahora podrá conectar dos nuevos valores a estos dos nodos.

Declarar un espacio de nombres predeterminado
Si desea declarar un espacio de nombres predeterminado, basta con asignar un valor de cadena vacío al nodo `prefix`. El resultado será (*fíjese en la parte resaltada*):

```
<?xml version="1.0" encoding="UTF-8"?>
<library xmlns="altova.library" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:noNamespaceSchemaLocation="Library.xsd">
...
```

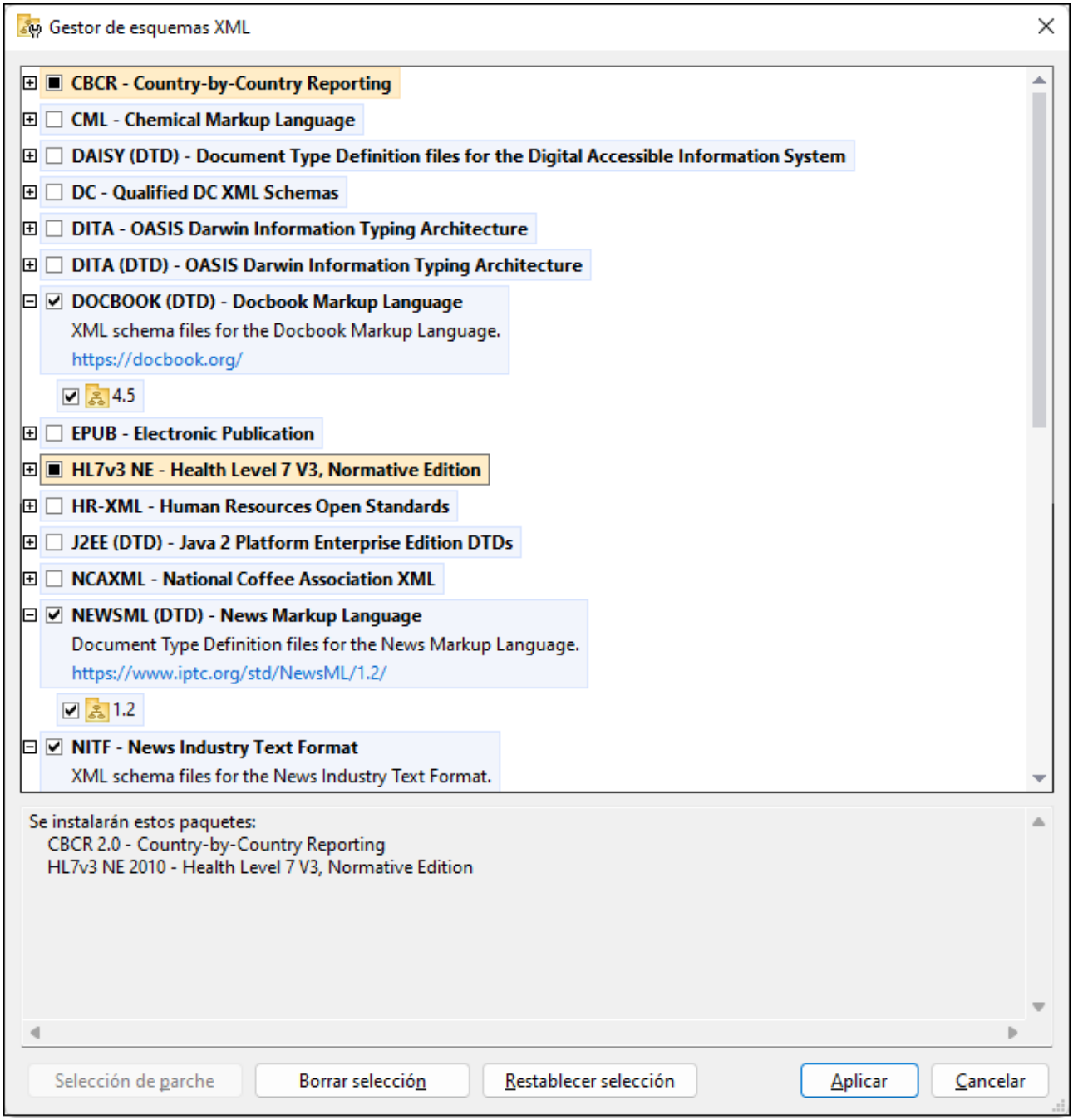
Si necesita crear prefijos para nombres de atributo (p. ej. `<number prod:id="prod557">557</number>`) puede habilitar el acceso dinámico a los atributos del nodo (*véase* *Asignar nombres de nodos* 385) o editar el esquema para que contenga un atributo `prod:id` para `<number>`.

Eliminar espacios de nombre
Para eliminar una declaración de espacio de nombres haga clic con el botón derecho en el nodo `ns:namespace` y seleccione **Quitar espacio de nombres** en el menú contextual.

4.1.8 Gestor de esquemas

El Gestor de esquemas XML es una herramienta que ofrece una forma centralizada de instalar y administrar esquemas XML (DTDs para XML y esquemas XML) para usarlos en todas las aplicaciones de Altova compatibles con XML Schema, incluido MapForce.

- En Windows, el gestor tiene una interfaz gráfica del usuario (*imagen siguiente*) a la que también puede acceder desde la línea de comandos. (Las aplicaciones de escritorio de Altova solo están disponibles para Windows; *consulte la lista siguiente*.)
- En Linux y macOS el Gestor de esquemas solo está disponible en la línea de comandos. (Las aplicaciones de escritorio de Altova están disponibles para Windows, Linux y macOS; *consulte la lista siguiente*.)



Aplicaciones de Altova que funcionan con el Gestor de esquemas

Aplicaciones de escritorio (solo para Windows)	Aplicaciones de servidor (Windows, Linux, macOS)
XMLSpy (todas las ediciones)	RaptorXML Server, RaptorXML+XBRL Server
MapForce (todas las ediciones)	StyleVision Server

StyleVision (todas las ediciones)	

Instalación y desinstalación del Gestor de esquemas

El Gestor de esquemas se instala automáticamente al instalar cualquiera de las aplicaciones de Altova compatibles con XML o el Altova Mission Kit (*véase la tabla de más arriba*).

También se elimina automáticamente si desinstala todas las aplicaciones de Altova compatibles con XML del equipo.

Características de Gestor de esquemas

El Gestor de esquemas permite:

- Ver los esquemas XML que hay instaladas en su equipo y comprobar si hay versiones nuevas para descargar.
- Descargar las versiones más recientes de los esquemas XML independientemente del ciclo de versiones de Altova. Altova guarda todos los esquemas en un sistema de almacenamiento en línea al que tiene acceso el Gestor de esquemas y desde donde puede descargarlas tan pronto como estén disponibles.
- Instalar o desinstalas cualquiera de las múltiples versiones de un esquema en concreto (o todas ellas, si las necesita).
- Un solo esquema XML representa un "paquete", pero puede tener dependencias en otros esquemas. Al instalar o desinstalar un esquema, se detectan e instalan o desinstalan también automáticamente todas sus dependencias. La interfaz gráfica del usuario (o la línea de comandos, en su caso) le informa cuando se añaden o eliminan esquemas.
- Los esquemas XML administradas con el Gestor de esquemas pueden usar el catálogo XML, que permite resolver referencias a URI en documentos de instancia o esquema desde archivos locales, en vez de a través de Internet.
- Todos los esquemas principales están incluidos en Gestor de esquemas y se actualizan de forma periódica a la versión más reciente. De esta forma puede administrar todos los esquemas desde un punto común y tenerlos siempre listos para las aplicaciones de Altova que las usan.
- Los cambios que se realizan en el Gestor de esquemas afectan a todos los productos de Altova que estén instalados en ese equipo.
- En los productos de Altova, si intenta validar con un esquema que no está instalado pero sí disponible con el Gestor de esquemas, este se instala automáticamente. Sin embargo, si el paquete de esquemas que quiere instalar contiene asignaciones de espacios de nombres, no puede instalarse automáticamente, sino que debe ejecutar Gestor de esquemas, seleccionar qué paquetes quiere instalar y ejecutar la instalación. Si después de instalar los paquetes la aplicación de Altova que está abierta no se reinicia automáticamente, debe reiniciarla manualmente.

Funcionamiento

Altova mantiene un almacenamiento en línea donde guarda todos los esquemas XML de los productos de Altova. Este almacenamiento se actualiza de forma periódica, por ejemplo, poco después de que las organizaciones correspondientes publiquen las versiones nuevas de los esquemas respectivos. Al ejecutar Gestor de esquemas desde la interfaz gráfica del usuario aparece información sobre los esquemas más recientes disponibles en un cuadro de diálogo en el que puede visualizarlos, instalarlos, actualizarlos o desinstalarlos.

También puede instalar los esquemas de otra manera. En el sitio web de Altova (https://www.altova.com/schema-manager) puede seleccionar el esquema y los esquemas dependientes de

este que quiere instalar. El sitio web prepara un archivo de tipo **.altova_xmlschemas** que puede descargar y que contiene la información sobre los esquemas seleccionados. Al hacer doble clic en este archivo o pasarlo a Gestor de esquemas desde la línea de comandos como argumento del comando **install**[141], Gestor de esquemas instala los esquemas que contiene.

Memoria caché local: seguimiento de esquemas

Independientemente de cómo se instalen los esquemas, toda la información sobre los esquemas instalados se almacena en una ubicación centralizada de su equipo, el directorio caché. El directorio caché local está en:

Windows	`C:\ProgramData\Altova\pkgs\.cache`
Linux	`/var/opt/Altova/pkgs\.cache`
macOS	`/var/Altova/pkgs`

El directorio caché local se actualiza automáticamente de vez en cuando para que el estado más actual del equipo corresponda con el del almacenamiento en línea. Más concretamente, el caché se actualiza:

- al ejecutar el Gestor de esquemas.
- al ejecutar MapForce por primera vez en un mismo día natural.
- si MapForce ya se está ejecutando, el directorio caché se actualiza cada 24 horas.
- también puede actualizar el caché local desde el almacenamiento en línea manualmente ejecutando el comando de actualización **update**[144] desde la línea de comandos.

Si instala o desinstala esquemas, el directorio caché local se actualiza automáticamente con información sobre los esquemas disponibles e instalados, además de con los propios archivos de esquema.

No modifique la memoria caché manualmente

El directorio caché local se mantiene automáticamente en base a los esquemas que instale o desinstale; no debe modificarlo ni eliminarlo manualmente. Si necesita restaurar el Gestor de esquemas a su estado original, ejecute el comando **reset**[142] desde la línea de comandos y después ejecute el comando **initialize**[140].

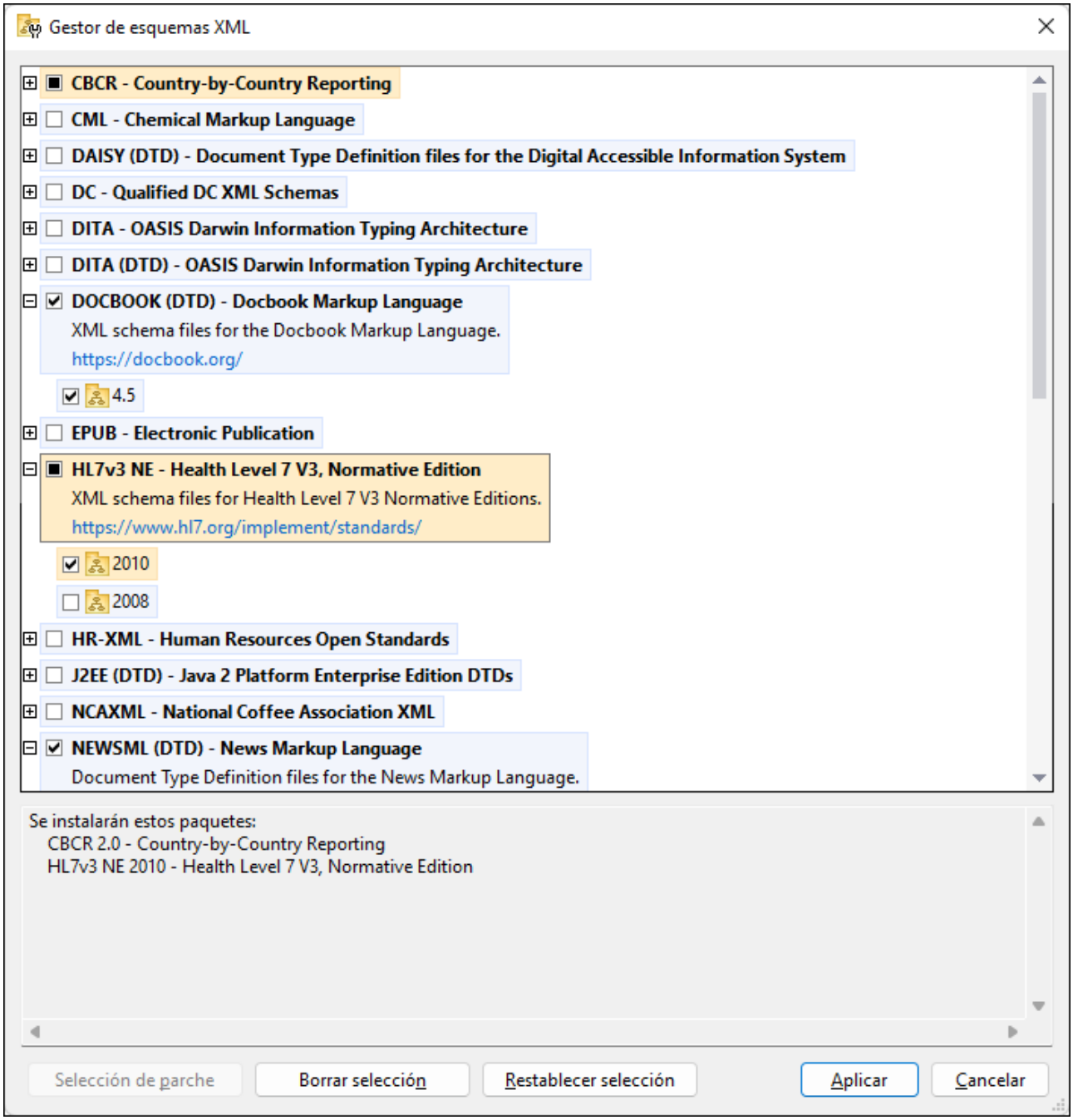
4.1.8.1 Ejecutar el gestor de esquemas

Interfaz gráfica del usuario

Hay varias formas de acceder a la IGU del Gestor de esquemas:

- *Durante la instalación de MapForce:* al final del proceso de instalación, seleccione la casilla *Invocar al Gestor de esquemas* para acceder directamente a la IGU del gestor de esquemas XML. Con él puede instalar esquemas durante el proceso de instalación de su aplicación de Altova.
- *Después de la instalación de MapForce:* una vez haya instalado la aplicación puede acceder al Gestor de esquemas en cualquier momento desde el comando de menú Herramientas | **Gestor de esquemas XML**.
- Mediante el archivo **.altova_schemas** que descargó del *sitio web de Altova:* haga doble clic en el archivo para ejecutar Gestor de esquemas, que instalará los esquemas que haya seleccionado.

Cuando se abra la IGU del Gestor de esquemas (*imagen siguiente*) podrá ver en ella los esquemas que ya se han instalado. Si quiere instalar más solo tiene que seleccionarlas, y al contrario si quiere desinstalar alguna. Una vez haya terminado, puede aplicar los cambios. los esquemas que se vayan a instalar o desinstalar aparecerán resaltados y un mensaje le avisará de los cambios que está a punto de hacer en la ventana *Mensajes*, en la parte inferior de la ventana de Gestor de esquemas (*véase imagen*).



Interfaz de la línea de comandos

Para ejecutar el Gestor de esquemas desde una interfaz de la línea de comandos debe usar su archivo ejecutable, **`xmlschemamanager.exe`**.

Puede encontrar este archivo:

- *en Windows:* `C:\ProgramData\Altova\SharedBetweenVersions`
- *en Linux o macOS (solo para aplicaciones de servidor):* `%INSTALLDIR%/bin`, donde `%INSTALLDIR%` es el directorio de instalación del programa.

Puede usar cualquiera de los comandos de la referencia de la línea de comandos 139, a continuación.

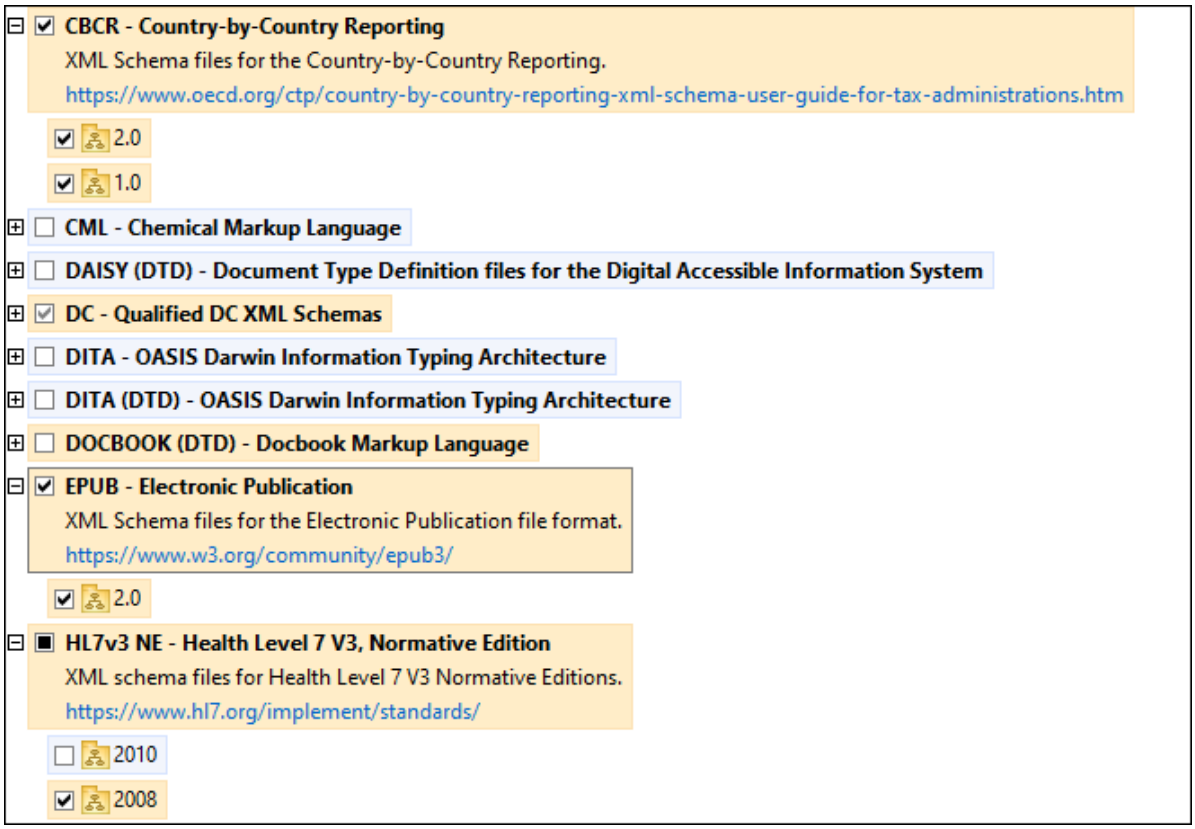
Para mostrar al ayuda de los comandos ejecute:

- *en Windows:* `xmlschemamanager.exe --help`
- *en Linux o macOS (solo para aplicaciones de servidor):* `sudo ./xmlschemamanager --help`

4.1.8.2 Categorías de estado

Gestor de esquemas diferencia los esquemas que administra entre:

- *Esquemas instalados:* estos aparecen en la IGU con sus casillas marcadas (*en la imagen siguiente las versiones marcadas de los esquemas EPUB y HL7v3 NE son las que están instalada*s). Si se seleccionan todas las versiones de un esquema, en la casilla del esquema aparece una marca de verificación. Si hay al menos un esquema sin seleccionar, en la en la casilla del esquema aparece un cuadrado negro. Para **desinstalar** un esquema debe desmarcar la casilla correspondiente (*en la imagen siguiente, el DTD DocBook está instalado y su casilla se ha desmarcado, es decir, se va a desinstalar*).
- *Esquemas disponibles no instalados:* estos aparecen en la IGU con las casillas correspondientes sin seleccionar. Para **instalar** esquemas, marque la casilla correspondiente.



- *Esquemas que pueden actualizarse:* son los que han sido revisados por sus emisores. Aparecen indicados en la IGU con el icono  (*imagen anterior*). Puede aplicar **parches** al esquema seleccionado con la revisión que esté disponible.

Puntos importantes

- En la imagen anterior se han marcado los esquemas CBCR. Las que tienen un fondo azul ya están instaladas. Las que tienen el fondo amarillo no están instaladas pero se han seleccionado para instalarlas. Observe que el esquema HL7v3 NE 2010 no está instalada ni se ha seleccionado para instalarlo.
- Al ejecutar el Gestor de esquemas desde la línea de comandos puede usar el comando **`list`** con distintas opciones para ver distintas categorías de esquemas:

`xmlschemamanager.exe list`	Muestra todas los esquemas instalados y disponibles; también indica qué esquemas se pueden actualizar
`xmlschemamanager.exe list -i`	Muestra solo los esquemas instalados; también indica qué esquemas se pueden actualizar
`xmlschemamanager.exe list -u`	Muestra qué esquemas se pueden actualizar

Nota: en Linux y macOS **`use sudo ./xmlschemamanager list`**

4.1.8.3 Aplicar parches o instalar un esquema

Aplicar un parche a un esquema instalado

A veces los emisores de los esquemas XML generan parches. Cuando el Gestor de esquemas XML detecta que hay parches disponibles, estos aparecen en las listas de esquemas, desde donde puede instalarlos.

En la IGU

Los parches se indican con el icono . (*Consulte también el apartado anterior sobre categorías de esquemas*[135].) Si hay parches disponibles se habilita el botón **Seleccionar parches**. Haga clic en él para seleccionar y preparar los parches. En la IGU, el icono de los esquemas correspondientes cambia de a y el cuadro de diálogo le informa de qué parches se van a aplicar. Las listas del panel principal y las del panel Mensajes están ordenadas alfabéticamente, por lo que puede ver y revisar los esquemas antes de aplicar los parches. Una vez esté listo para instalar los parches seleccionados, haga clic en **Aplicar**.

En la línea de comandos

Para aplicar un parche desde lal ínea de comandos:

1. Ejecute el comando `list -u`[142]. Aparece una lista con los esquemas para las que hay parches disponibles.
1. Ejecute el comando `upgrade`[145] para instalar todos los parches.

Instalar un esquema disponible

Para instalar esquemas puede usar la IGU del Gestor de esquemas o enviar las instrucciones al Gestor de esquemas desde la línea de comandos.

Nota: si el esquema actual tiene dependencias en otros esquemas, también se instalan (o desinstalan, según el caso) los esquemas dependientes.

En la IGU

Para instalar esquemas con la IGU del Gestor de esquemas, seleccione los esquemas que quiere instalar y haga clic en **Aplicar**.

También puede seleccionar los esquemas que quiere instalar en el sitio web de Altova y generar desde allí un archivo **`.altova_schemas`**. Al hacer doble clic en este archivo se abre el Gestor de esquemas con los esquemas que indicó preseleccionados. Solo tiene que hacer clic en **Aplicar**.

En la línea de comandos

Para instalar esquemas desde la línea de comandos ejecute el comando **`install`**:

```
xmlschemamanager.exe install [opciones] Schema+
```

donde **`FILTER`** es el esquema (o los esquemas) que quiere instalar o un archivo **`.altova_schemas`**. Para hacer referencia a un esquema se usa un identificados con el formato **`<nombre>-<versión>`** que aparece junto a cada esquema que muestra el comando `list`[142]. Puede introducir tantos esquemas como quiera. Para más detalles consulte la descripción del comando `install`[141].

Nota: en Linux o macOS, use el comando **`sudo ./xmlschemamanager`**.

Instalar un esquema requerido
Si ejecuta un comando en MapForce y MapForce descubre que uno de los esquemas que necesita para ejecutar el comando falta o está incompleta, el Gestor de esquemas incluirá información sobre ese componente de esquema que falta. Entonces puede aplicar el parche indicado y/o instalar el esquema que falta.

Siempre puede ver todos los esquemas instalados previamente ejecutando el Gestor de esquemas desde **Herramientas | Gestor de esquemas**.

4.1.8.4 Desinstalar o restaurar esquemas

Desinstalar un esquema

Para desinstalar esquemas puede usar la IGU del Gestor de esquemas o enviar las instrucciones al Gestor de esquemas desde la línea de comandos.

Nota: si el esquema actual tiene dependencias en otros esquemas, también se instalan (o desinstalan, según el caso) los esquemas dependientes.

En la IGU
Para desinstalar esquemas con la IGU del Gestor de esquemas, seleccione los esquemas que quiere desinstalar y haga clic en **Aplicar**. Los esquemas seleccionadas y sus dependencias se desinstalarán.

Para desinstalar todos los esquemas haga clic en **Deseleccionar todas** y haga clic en **Aplicar**.

En la línea de comandos
Para desinstalar esquemas desde la línea de comandos ejecute el comando `uninstall`[143]:

```
xmlschemamanager.exe uninstall [options] FILTER+
```

donde **FILTER** es el esquema (o los esquemas) que quiere desinstalar o un archivo **.altova_schemas**. Para hacer referencia a un esquema se usa un identificador con el formato **<nombre>-<versión>** que aparece junto a cada esquema que muestra el comando `list`[142]. Puede introducir tantos esquemas como quiera. Para más detalles consulte la descripción del comando `uninstall`[143].

Nota: en Linux o macOS, use el comando **sudo ./xmlschemamanager**.

Restaurar el Gestor de esquemas

Puede restaurar el Gestor de esquemas, es decir, eliminar todos los esquemas instaladas, así como el directorio caché.

- En la IGU, haga clic en **Restaurar selección**.
- En la línea de comandos, use el comando `reset`[142].

Una vez haya ejecutado este comando, asegúrese de que ejecuta también el comando `initialize`[140] para recrear el directorio caché. También puede ejecutar el comando `reset`[142] con la opción `-i`.

Recuerde que `reset -i`[142] restaura la instalación original del producto, por lo que es recomendable ejecutar el comando `update`[144] después de restaurar el gestor. Puede ejecutar el comando `reset`[142] con las opciones `-i` o `-u`.

4.1.8.5 Interfaz de la línea de comandos (ILC)

Para llamar a Gestor de esquemas desde la línea de comandos necesita saber la ruta del ejecutable. Por defecto, el ejecutable del Gestor de esquemas se encuentra en:

```
C:\ProgramData\Altova\SharedBetweenVersions\XMLSchemaManager.exe
```

Nota: en los sistemas Linux y macOS una vez haya cambiado el directorio al que contiene el ejecutable, puede llamar al ejecutable con **`sudo ./xmlschemamanager`**.El prefijo **`./`** indica que el ejecutable está en el directorio actual. El prefijo **`sudo`** indica que el comando se debe ejecutar con derechos de administrador.

Sintaxis de la línea de comandos

La sintaxis general para usar la línea de comandos es:

```
<exec> -h | --help | --version | <command> [opciones] [argumentos]
```

En el código anterior la barra vertical **`|`** separa elementos que se excluyen mutuamente. Los corchetes **`[]`** indican elementos opcionales. Básicamente, puede teclear la ruta del ejecutable seguida por las opciones `--h`, `--help` o `--version`, o por un comando. Cada comando puede tener opciones y argumentos. Los comandos se describen en los apartados siguientes.

4.1.8.5.1 help

Este comando ofrece ayuda contextual sobre los comandos del ejecutable del Gestor de esquemas.

Sintaxis

```
<exec> help [command]
```

Donde `[command]` es un argumento opcional que indica cualquier nombre válido de comando.

Tenga en cuenta que:

- Puede invocar la ayuda tecleando un comando seguido por `--h` or `--help`, por ejemplo: **`<exec> list -h`**
- Puede invocar la ayuda general tecleando `--h` o `--help` directamente después del ejecutable, por ejemplo:

Ejemplo

Este comando muestra la ayuda del comando `list`:

```
xmlschemamanager help list
```

4.1.8.5.2 info

Este comando muestra información detallada sobre cada uno de los esquemas dados como argumento. Esta información incluye el título, la versión, la descripción, el editor y las referencias de las dependencias.

Sintaxis

```
<exec> info [options] Schema+
```

- El argumento **`Schema`** es el nombre de un esquema o parte del nombre de un esquema. (Para ver el ID de un paquete de esquemas y la información relativa a su estado de instalación use el comando `list`[142].)
- Use **`<exec> info -h`** para ver la ayuda sobre este comando en la línea de comandos..

Ejemplo

Este comando muestra información detallada sobre los esquemas **`DocBook-DTD`** y **`NITF`**:

```
xmlschemamanager info doc nitf
```

4.1.8.5.3 initialize

Este comando inicializa el entorno del Gestor de esquemas y crea un directorio caché donde se guardan todos los esquemas localmente. El Gestor de esquemas se inicializa automáticamente la primera vez que instale una aplicación de Altova compatible con él, por lo que normalmente no es necesario ejecutar este comando. Por lo general solo es necesario ejecutarlo después de haber ejecutado el comando `reset`.

Sintaxis

```
<exec> initialize | init [opciones]
```

Opciones
Estas son las opciones del comando **`initialize`**:

`--help, --h`	Muestra la ayuda sobre este comando en la línea de comandos.
`--silent, --s`	Muestra solamente los mensajes de error. El valor predeterminado es **`false`**.

`--verbose, --v`	Muestra información suplementaria durante la ejecución. El valor predeterminado es **`false`**.

Ejemplo

Este comando inicializa el Gestor de esquemas:

```
xmlschemamanager initialize
```

4.1.8.5.4 install

Este comando instala una o más esquemas.

Sintaxis

```
<exec> install [options] Schema+
```

Para indicar varios esquemas, repita el argumento `Schema` tantas veces como sea necesario.

El argumento de `Schema` puede ser:

1. Un identificador de esquema en el formato **`<name>-<version>`**, por ejemplo: **`cbcr-2.10`**). Para ver todos los identificadores de esquemas y sus versiones ejecute el comando `list`[142]. También puede usar el nombre de el esquema abreviado, si este es único, por ejemplo **`docbook`**. Si usa una abreviación del nombre se desinstalan todos los esquemas que contengan esa abreviación.
2. La ruta de acceso a un archivo **`.altova_schemas`** descargado desde el sitio web de Altova. Para más información sobre estos archivos consulte la *Introducción al Gestor de esquemas: funcionamiento*[130].

Opciones

Estas son las opciones del comando **`install`**:

`--help, --h`	Muestra la ayuda sobre este comando en la línea de comandos.
`--silent, --s`	Muestra solamente los mensajes de error. El valor predeterminado es **`false`**.
`--verbose, --v`	Muestra información suplementaria durante la ejecución. El valor predeterminado es **`false`**.

Ejemplo

Este comando instala el esquema CBCR 2.0 (Country-By-Country Reporting) y el DTD DocBook más reciente:

```
xmlschemamanager install cbcr-2.0 docbook
```

4.1.8.5.5 list

Use este comando para ver los esquemas del Gestor de esquemas; tiene varias opciones:

- lista de todos los esquemas disponibles
- lista de esquemas específicos
- lista de los esquemas instalados
- lista de los esquemas que se pueden actualizar.

Sintaxis

```
<exec> list | ls [options] Schema?
```

Si no se indica ningún argumento **Schema** la lista incluye todos los esquemas. De lo contrario la lista incluye los esquemas indicados en las opciones (*véase el ejemplo de más abajo*). Recuerde que puede usar el argumento **Schema** tantas veces como quiera.

Opciones
Estas son las opciones del comando **list**:

`--help, --h`	Muestra la ayuda sobre este comando en la línea de comandos.
`--installed, --i`	Muestra solamente los esquemas instaladas. El valor predeterminado es **false**.
`--upgradeable, --u`	Muestra solamente los esquemas para las que hay disponible una versión más reciente (parches). El valor predeterminado es **false**.

Ejemplos

- Para ver todos los esquemas disponibles ejecute: **xmlschemamanager list**
- Para ver solamente los esquemas instaladas ejecute: **xmlschemamanager list -i**
- Para ver todos los esquemas cuyos nombres contienen "doc" o "nitf" ejecute: **xmlschemamanager list doc nitf**

4.1.8.5.6 reset

Este comando elimina todos los esquemas instalados, así como el directorio de caché. Este comando elimina todos los esquemas instalados y su información. Una vez haya ejecutado este comando, asegúrese de que ejecuta el comando `initialize` 140 para volver a crear el directorio de caché. También puede ejecutar el comando `reset` con la opción `-i`. Tenga en cuenta que `reset -i` restaura la instalación original del producto, por lo que se recomienda ejecutar también el comando `update` 144 después de una restauración. También puede ejecutar el comando `reset` con las opciones `-i` y `-u`.

Sintaxis

```
<exec> reset [opciones]
```

Opciones
Estas son las opciones del comando **reset**:

`--help, --h`	Muestra la ayuda sobre este comando en la línea de comandos.
`--init, --i`	Inicializa el entorno del Gestor de esquemas XML después de una restauración. El valor predeterminado es **false**.
`--silent, --s`	Muestra solamente los mensajes de error. El valor predeterminado es **false**.
`--update, --u`	Inicializa y actualiza el entorno del Gestor de esquemas XML después de una restauración. El valor predeterminado es **false**.
`--verbose, --v`	Muestra información suplementaria durante la ejecución. El valor predeterminado es **false**.

Ejemplos

- Para restaurar el Gestor de esquemas, ejecute: **xmlschemamanager reset**
- Para restaurar el Gestor de esquemas e inicializarlo, ejecute: **xmlschemamanager reset -i**
- Para restaurar el Gestor de esquemas, inicializarlo y actualizar la lista de esquemas, ejecute: **xmlschemamanager reset -i -u**

4.1.8.5.7 uninstall

Este comando desinstala una o más esquemas. Por defecto, cualquier esquema a la que haga referencia el esquema actual también se desinstala. Para desinstalar solamente el esquema actual y mantener aquellas a las que se hace referencia, use la opción `--k`.

Sintaxis

```
<exec> uninstall [opciones] Schema+
```

Para indicar varios esquemas, repita `FILTER` tantas veces como sea necesario.

El argumento de **Schema** puede ser:

- Un identificador de esquema en el formato **<name>-<version>**, por ejemplo: **eba-2.10**). Para ver todos los identificadores de esquemas y sus versiones ejecute el comando `list -i`[142]. También puede usar el nombre del esquema abreviado, si este es único, por ejemplo **eba**. Si usa una abreviación del nombre se desinstalan todos los esquemas que contengan esa abreviación.
- La ruta de acceso a un archivo **.altova_taxonomies** descargado desde el sitio web de Altova. Para más información sobre estos archivos consulte la *Introducción al Gestor de esquemas: funcionamiento*[130].

Opciones
Estas son las opciones del comando **uninstall**:

`--help, --h`	Muestra la ayuda sobre este comando en la línea de comandos.
`--keep-references, --k`	Si usa esta opción, los esquemas referenciados no se desinstalan. El valor predeterminado es **false**.
`--silent, --s`	Muestra solamente los mensajes de error. El valor predeterminado es **false**.
`--verbose, --v`	Muestra información suplementaria durante la ejecución. El valor predeterminado es **false**.

Ejemplo

Este comando desinstala los esquemas CBCR 2.0 y EPUB 2.0:

```
xmlschemamanager uninstall cbcr-2.0 epub-2.0
```

Este comando desinstala el esquema **eba-2.10** pero no los esquemas a los que hace referencia:

```
xmlschemamanager uninstall --k cbcr-2.0
```

4.1.8.5.8 update

Este comando consulta la lista de esquemas disponibles en el almacenamiento en línea y actualiza el directorio de caché local. Esta información se actualiza de forma implícita, por lo que no es necesario ejecutar este comando a no ser que haya ejecutado `reset`[142] e `initialize`[140].

Sintaxis

```
<exec> update [opciones]
```

Opciones
Estas son las opciones del comando `update`:

`--help, --h`	Muestra la ayuda sobre este comando en la línea de comandos.
`--silent, --s`	Muestra solamente los mensajes de error. El valor predeterminado es **false**.
`--verbose, --v`	Muestra información suplementaria durante la ejecución. El valor predeterminado es **false**.

Ejemplo

Este comando actualiza la lista de esquemas:

```
xmlschemamanager update
```

4.1.8.5.9 upgrade

Este comando actualiza todos los esquemas aptos para la versión *parche* más reciente disponible. Puede identificar cuáles lo son con el comando `list -u`.

Nota: el comando `upgrade` eliminaría un esquema obsoleto si no hay ninguna versión disponible.

Sintaxis

```
<exec> upgrade [opciones]
```

Opciones
Estas son las opciones del comando `upgrade`:

`--help, --h`	Muestra la ayuda sobre este comando en la línea de comandos.
`--silent, --s`	Muestra solamente los mensajes de error. El valor predeterminado es **`false`**.
`--verbose, --v`	Muestra información suplementaria durante la ejecución. El valor predeterminado es **`false`**.

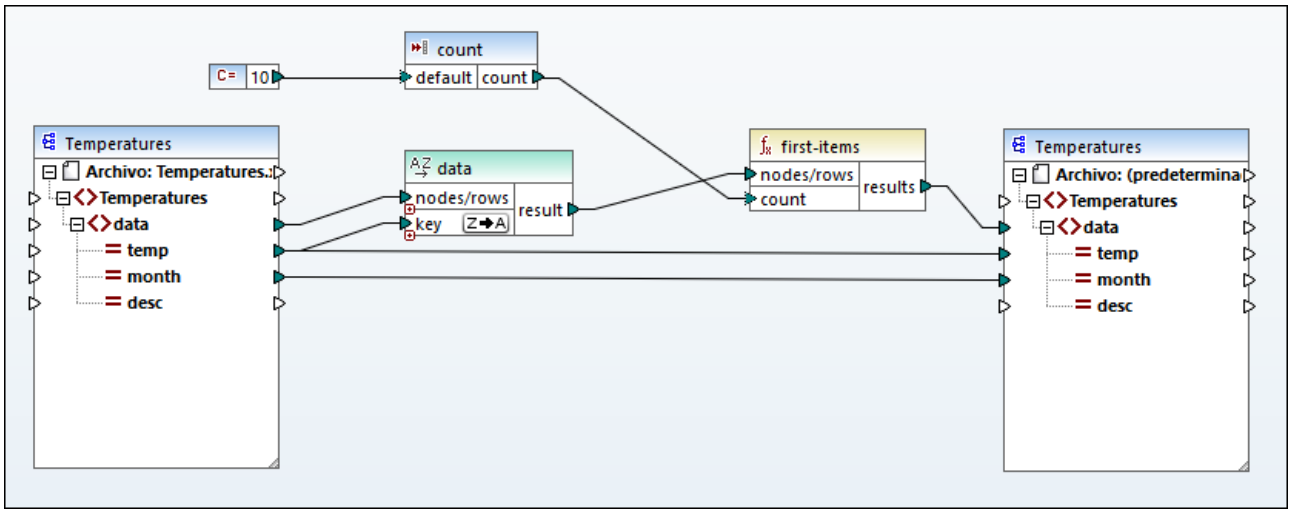
5 Componentes de transformación

En esta sección explicamos los distintos componentes que se pueden usar para transformar datos o para almacenarlos de forma temporal para seguir procesándolos más adelante. A continuación puede ver una lista de los componentes de transformación:

Tenga en cuenta que las funciones también pertenecen a los componentes de transformación. Sin embargo, hemos agrupado esas funciones 196 en una sección propia.

5.1 Entrada simple

Si tiene que crear una asignación que tome parámetros como componentes de entrada puede hacerlo añadiendo un componente especial llamado "componente de entrada simple". Este tipo de componente siempre tiene un tipo de datos simple (por ejemplo, cadena, número entero, etc.) en lugar de una estructura de elementos y secuencias. Por ejemplo, en la asignación siguiente encontramos el componente de entrada simple **count**, que sirve para dar como parámetro el número máximo de filas que debe recuperar el archivo XML de entrada (cuyo valor predeterminado es **10**). Es importante recordar que los nodos que se indican como componente de entrada a la función first-items [283] se ordenan con ayuda de un componente de ordenación, por lo que la asignación da como resultado solamente las *N* temperaturas más altas, donde *N* es el valor del parámetro.



FindHighestTemperatures.mfd

Otro uso bastante común de los componentes de entrada de tipo simple es dar un nombre de archivo a la asignación. Esto puede ser útil para asignaciones que lean los archivos de entrada o escriban los archivos de salida de forma dinámica (véase Procesar varios archivos de entrada o salida simultáneamente [402]).En el archivo XSLT que se genera los componentes de entrada simples corresponden a parámetros de hojas de estilos.

Cada componente de entrada simple (o parámetro) se puede crear como componente opcional u obligatorio (*véase Configurar componentes de entrada simples* [149]). Si fuera necesario, también podrá crear valores predeterminados para los parámetros de entrada de la asignación (*véase Crear un valor de entrada predeterminado* [150]). Esto le permitirá ejecutar la asignación de forma segura aunque no aporte explícitamente un valor de parámetro en tiempo de ejecución de la asignación.

Los parámetros de entrada añadidos en el área de asignación principal no se deben confundir con los parámetros de entrada de las funciones definidas por el usuario (*véase Funciones definidas por el usuario* [205]). En esta tabla puede ver en qué se parecen y en qué se diferencian:

Parámetros de entrada de la asignación	**Parámetros de entrada de funciones definidas por el usuario**

Se añaden con el comando **Función | Insertar componente de entrada**.	Se añaden con el comando **Función | Insertar componente de entrada**.
Pueden tener tipos de datos simples (cadena, entero, etc.).	Puede tener tipos de datos simples y complejos.
Afectan a toda la asignación.	Afectan sólo al contexto de la función donde se definen.

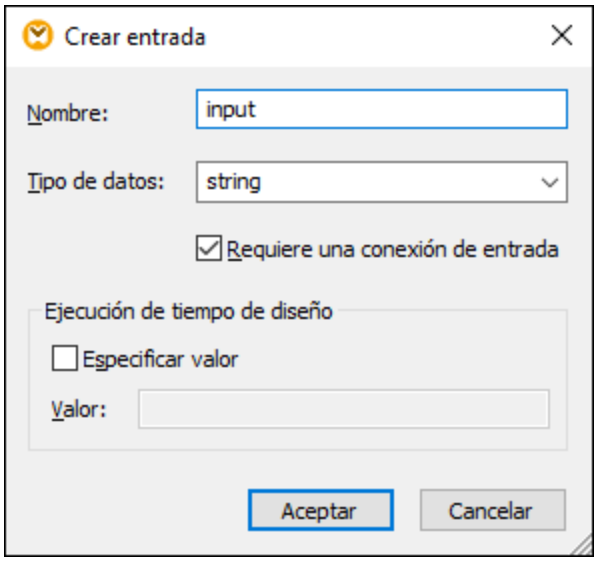
Cuando cree una asignación invertida (con el comando de menú **Herramientas | Crear asignación inversa**), los componentes de entrada simples se convertirán en componentes de salida simples.

5.1.1 Agregar componentes de entrada simples

Para agregar un componente de entrada simple a la asignación:

1. Asegúrese de que en la ventana de asignación está activa la asignación principal (y no una función definida por el usuario).
2. Elija una opción:

   - En el menú **Función** haga clic en **Insertar componente de entrada**.
   - En el menú **Insertar** haga clic en **Insertar componente de entrada**.
   - Haga clic en el botón de la barra de herramientas **Insertar componente de entrada**.



3. Introduzca un nombre y seleccione el tipo de datos que requiere esta entrada. Si la entrada debe tratarse como un parámetro obligatorio de la asignación, marque la casilla Requiere una conexión de entrada. (Para más información sobre las demás opciones de configuración consulte el apartado Configurar componentes de entrada simples 149.)

Nota: El nombre del parámetro sólo puede contener letras, cifras y barra baja; el resto de caracteres no están permitidos. Esto permite que las asignaciones funcionen con todos los lenguajes de generación de código.

4. Haga clic en **Aceptar**.

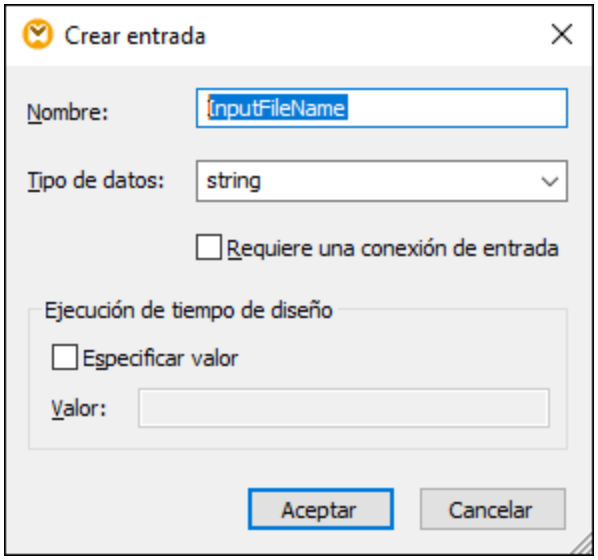
Las demás opciones de configuración se describen en el apartado Configurar componentes de entrada simples 149.

5.1.2 Configurar componentes de entrada simples

Los componentes de entrada simples se pueden configurar en el momento en el que se añaden al área de asignación o más tarde. Esto se hace en el cuadro de diálogo "Crear entrada".

Para abrir el cuadro de diálogo "Crear entrada" tiene tres opciones:

- Seleccione el componente y seleccione el comando de menú **Componente | Propiedades**.
- Haga doble clic en el componente.
- Haga clic con el botón derecho en el componente y después elija **Propiedades** en el menú contextual.



Cuadro de diálogo "Crear entrada"

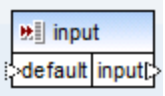
Estas son las opciones de configuración que ofrece el cuadro de diálogo:

Nombre	Introduzca un nombre descriptivo para el parámetro de entrada que corresponde a este componente. En tiempo de ejecución el valor que se introdujo en este campo pasa a ser el nombre del parámetro que se pasa a la asignación. Por tanto, no está permitido el uso de espacios ni caracteres especiales.

Tipo de datos	Todos los parámetros de entrada se tratan por defecto como tipo de datos de cadena. Si el parámetro debe tener otro tipo de datos, seleccione el valor pertinente en la lista. Cuando la asignación se ejecute, MapForce convertirá el tipo del parámetro de entrada en el tipo de datos que se seleccionó aquí.
Requiere una conexión de entrada	Si marca esta casilla, el parámetro de entrada será obligatorio (es decir, la asignación no se podrá ejecutar a no ser que se aporte un valor de parámetro). Desactive esta casilla si desea especificar un valor predeterminado para el parámetro de entrada (*véase* *Crear un valor de entrada predeterminado* 150).
Especificar valor	Esta casilla sólo es relevante cuando se ejecuta la asignación en tiempo de diseño (cuando se hace clic en el panel *Resultados*). Permite introducir directamente en el componente el valor que se debe usar como entrada de la asignación.
Valor	Este campo sólo es relevante cuando se ejecuta la asignación en tiempo de diseño (cuando se hace clic en el panel *Resultados*). Para introducir el valor que debe usar MapForce como entrada de la asignación, marque la casilla *Especificar valor* y después escriba el valor en este campo. **Nota:** si marca la casilla *Especificar valor* e introduce un valor en el campo adyacente, ese valor tendrá preferencia frente al valor predeterminado en la vista previa de la asignación (es decir, en el momento de diseñarla). Sin embargo, este valor no afecta al código XSLT, XQuery o de programa, a la ejecución de MapForce Server o a la implementación en FlowForce Server.

5.1.3 Crear un valor de entrada predeterminado

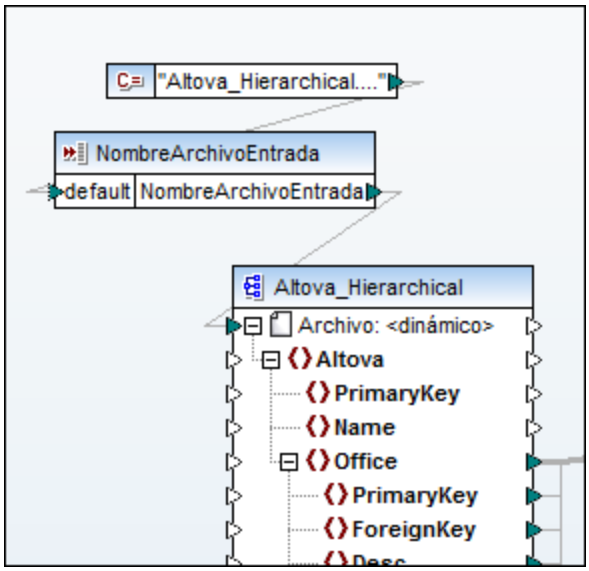
Tras añadir el componente de entrada al área de asignación, debemos fijarnos en el elemento **`default`** situado en el lado izquierdo del componente.



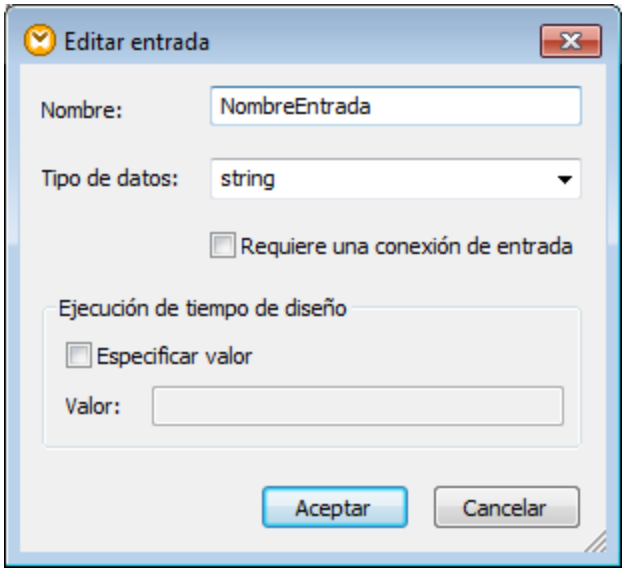
Componente de entrada simples

El elemento **`default`** permite conectar un valor predeterminado opcional a este componente de entrada. Esto se hace de la siguiente manera:

1. Añada un componente de constante (con el comando de menú **Insertar | Constante**) y después conéctelo con el elemento **`default`** del componente de entrada.



2. Haga doble clic en el componente de entrada y compruebe que la casilla *Requiere una conexión de entrada* está desactivada. Cuando se crea un valor de entrada predeterminado, esta casilla no es relevante y causa advertencias de validación.



3. Haga clic en **Aceptar** para terminar.

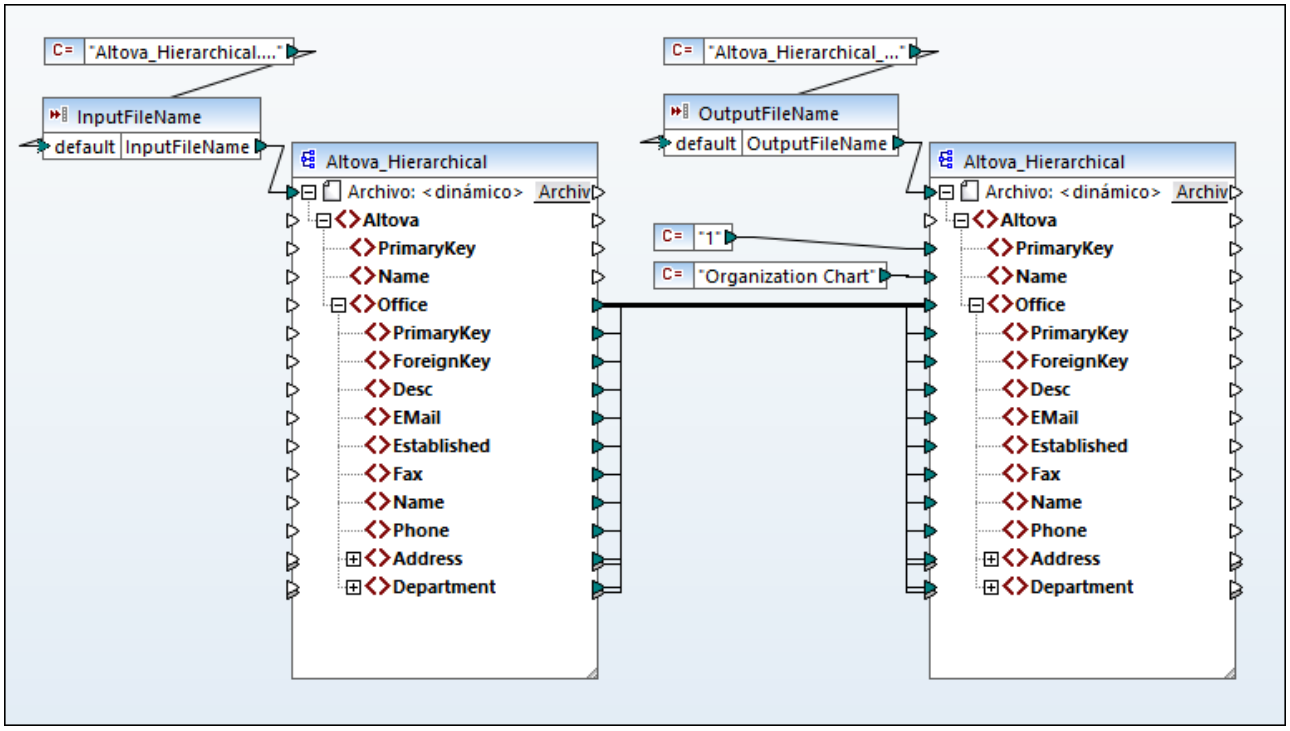
Nota: si marca la casilla *Especificar valor* e introduce un valor en el campo adyacente, ese valor tendrá preferencia frente al valor predeterminado en la vista previa de la asignación (es decir, en el momento de diseñarla). Sin embargo, este valor no afecta al código XSLT, XQuery o de programa, a la ejecución de MapForce Server o a la implementación en FlowForce Server.

5.1.4 Ejemplo: usar nombres de archivo como parámetros de asignación

En este ejemplo se explica paso a paso cómo ejecutar una asignación que toma parámetros de entrada en tiempo de ejecución. El archivo de diseño de asignación que utilizamos en este ejemplo está en `<Documentos>\Altova\MapForce2024\MapForceExamples\FileNamesAsParameters.mfd`.

Esta asignación lee datos de un archivo XML de origen y los escribe en un archivo XML de destino. Los datos se escriben en el archivo de destino prácticamente sin cambios (los atributos `PrimaryKey` y `Name` se rellenan con valores constantes de la asignación). El objetivo principal de la asignación es permitir que se pueda indicar el nombre de los archivos de entrada y salida como parámetros al ejecutar la asignación.

La asignación utiliza dos componentes de entrada: **InputFileName** y **OutputFileName**. Estos componentes aportan el nombre de archivo de entrada (y el nombre de archivo de salida respectivamente) del archivo XML de origen y de destino. Por este motivo, están conectados con el nodo `Archivo: <dinámico>`. Puede cambiar un componente a este modo haciendo clic en el botón **Archivo** (Archivo) y seleccionando **Usar nombres de archivo dinámicos dados por la asignación**.



FileNamesAsParameters.mfd (MapForce Basic Edition)

Si hace doble clic en la barra del título de uno de los componentes (**InputFileName** u **OutputFileName**), puede ver o editar sus propiedades. Por ejemplo, puede indicar el tipo de datos del parámetro de entrada o cambiar el nombre del parámetro de entrada, como se explica en Configurar componentes de entrada simples 149. En este ejemplo los parámetros de entrada y salida se han configurado así:

- El parámetro **InputFileName** es de tipo "cadena" y tiene un valor predeterminado dado por una constante definida en la misma asignación. Esa constante es de tipo "cadena" y su valor es

"Altova_Hierarchical.xml". Por tanto, al ejecutar la asignación, esta intentará leer datos de un archivo llamado "Altova_Hierarchical.xml", siempre que no indique otro valor como parámetro.
- El parámetro **OutputFileName** es de tipo "cadena" y su valor predeterminado también viene dado por una constante definida en la misma asignación. Esa constante también es de tipo "cadena" y se llama "Altova_Hierarchical_output.xml", por lo que al ejecutar la asignación esta genera un archivo XML de salida llamado "Altova_Hierarchical_output.xml", siempre que no indique otro valor como parámetro.

Ambos componentes (**InputFileName** y **OutputFileName**) son componentes de entrada simples, así que podrá suministrarlos como parámetros de entrada a la hora de ejecutar la asignación. A continuación explicamos cómo hacerlo en estos lenguajes de transformación:

- XSLT 2.0 153, con ayuda de RaptorXML Server

XSLT 2.0

Si genera código en XSLT 1.0, XSLT 2.0 o XSLT 3.0, los parámetros de entrada se escriben en el archivo por lotes `DoTransform.bat` que se ejecutará con RaptorXML Server (*véase Automatización con RaptorXML Server* 428). Para usar otro archivo de entrada (o de salida), puede pasar los parámetros necesarios en la línea de comandos cuando llame al archivo `DoTransform.bat` o editar este último para incluir los parámetros necesarios.

Siga estos pasos para suministrar un parámetro de entrada personalizado al archivo `DoTransform.bat`:

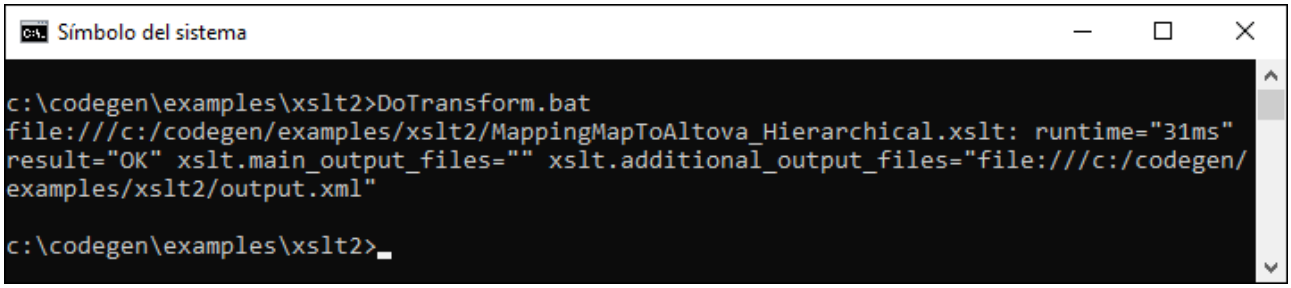
1. Primero genere código XSLT. Por ejemplo, genere 2.0 (**Archivo | Generar código en | XSLT 2.0**) a partir del diseño de asignación `FileNamesAsParameters.mfd`.

2. Copie el archivo `Altova_Hierarchical.xml` del directorio `<Documentos>\Altova\MapForce2024\MapForceExamples\` al directorio donde generó el código XSLT 2.0 (en este ejemplo usamos el directorio de salida `c:\codegen\examples\xslt2\`). Este archivo hará las veces de parámetro personalizado.

3. Edite el archivo por lotes `DoTransform.bat` para incluir el parámetro de entrada personalizado antes o después de `%*` (ver texto resaltado más abajo). Observe que el valor de parámetro va entre comillas simples. Los parámetros de entrada disponibles se enumeran en la sección **rem** (Remark).

```
@echo off

RaptorXML xslt --xslt-version=2 --input="MappingMapToAltova_Hierarchical.xslt" --
param=InputFileName:'Altova_Hierarchical.xml' %*
"MappingMapToAltova_Hierarchical.xslt"
rem --param=InputFileName:
rem --param=OutputFileName:
IF ERRORLEVEL 1 EXIT/B %ERRORLEVEL%
```

Cuando ejecute el archivo por lotes `DoTransform.bat`, RaptorXML Server finaliza la transformación usando `Altova_Hierarchical.xml` como parámetro de entrada.



C#

Para indicar un parámetro de entrada personal a la aplicación línea de comandos en C# generada por MapForce:

1. Abra el ejemplo **FileNamesAsParameters.mfd** que encontrará en el directorio **<Documentos>\Altova\MapForce2024\MapForceExamples**.
2. En el menú **Archivo** haga clic en **Generar código en | C#** y seleccione un directorio de destino (que en este ejemplo es **C:\codegen\examples\cs**).
3. Abra la solución en Visual Studio y pulse **Ctrl + Mayús + B** para generarla.
4. Copie el archivo **Altova_Hierarchical.xml** desde **<Documentos>\Altova\MapForce2024\MapForceExamples** al directorio en el que se genera **Mapping.exe** (en este ejemplo, **C:\codegen\examples\cs\Mapping\bin\Debug**). Como hemos explicado anteriormente, la asignación intentará leer este archivo si no indica otro valor para el parámetro **InputFileName**.
5. Abra una ventana de la línea de comandos y cambie al directorio en el que se encuentra **Mapping.exe**.

   ```
   cd C:\codegen\examples\cs\Mapping\bin\Debug
   ```

6. Ejecute la aplicación con este comando:

   ```
   Mapping.exe /OutputFileName output.xml
   ```

En el comando anterior el parámetro de entrada `/OutputFileName` indica el nombre del archivo de salida que se va a generar.

C++

Para indicar un un parámetro de entrada personal a la aplicación línea de comandos en C++ generada por MapForce:

1. Abra el ejemplo **FileNamesAsParameters.mfd** que encontrará en el directorio **<Documentos>\Altova\MapForce2024\MapForceExamples**.
7. En el menú **Archivo** haga clic en Generar código en **Generar código en | C++** y seleccione un directorio de destino (que en este ejemplo es **C:\codegen\examples\cpp**).
2. Abra la solución en Visual Studio y pulse **Ctrl + Mayús + B** para generarla.
3. Copie el archivo **Altova_Hierarchical.xml** desde **<Documentos>\Altova\MapForce2024\MapForceExamples** al directorio en el que se genera

Mapping.exe (en este ejemplo **C:\codegen\examples\cpp\Mapping\Debug**). Como hemos explicado anteriormente, la asignación intentará leer este archivo si no indica otro valor para el parámetro **InputFileName**.

8. Abra una ventana de la línea de comandos y cambie al directorio en el que se encuentra **Mapping.exe**.

```
cd C:\codegen\examples\cpp\Mapping\Debug
```

4. Ejecute la aplicación con este comando:

```
Mapping.exe /OutputFileName output.xml
```

En el comando anterior el parámetro de entrada `/OutputFileName` indica el nombre del archivo de salida que se va a generar.

5.2 Salida simple

Cuando necesite devolver un valor de cadena de una asignación, puede usar un componente de salida simple. En el área de asignación los componentes de salida simples desempeñan el papel de componente de destino que tiene un tipo de datos de cadena en lugar de una estructura de elementos y secuencias. Por tanto, puede crear un componente de salida simple en lugar de (o además de) un componente de destino basado en un archivo. Por ejemplo, puede usar un componente de salida simple para probar y obtener una vista previa del resultado de una función (*véase* *Ejemplo: vista previa de resultados de una función* 157).

Los componentes de entrada simples no se deben confundir con los parámetros de salida de las funciones definidas por el usuario (*véase* *Funciones definidas por el usuario* 205). En esta tabla puede ver en qué se parecen y en qué se diferencian:

Componentes de salida	**Parámetros de salida de funciones definidas por el usuario**
Se añaden desde el comando de menú **Función | Insertar componente de salida**.	Se añaden desde el comando de menú **Función | Insertar componente de salida**.
Tienen el tipo de datos "string".	Puede tener tipos de datos simples y complejos.
Afectan a toda la asignación.	Afectan sólo al contexto de la función donde se definen.

Si lo necesita, puede añadir varios componentes de salida simples a la asignación. También puede usar componentes de salida simples junto con componentes de destino basados en archivos. Si la asignación contiene varios componentes de destino, puede consultar la vista previa de los datos que devuelve cada componente haciendo clic en el botón **Vista previa** () de la barra de título del componente correspondiente y haciendo clic en el panel *Resultados* de la ventana de asignación.

Los componentes de salida simples se pueden usar en estos lenguajes de transformación de MapForce:

Lenguaje	**Funcionamiento**
XSLT 1.0, XSLT 2.0, XSLT 3.0	En los archivos XSLT que se generan, el componente de salida simple definido en la asignación se convierte en el resultado de la transformación XSLT. Si usa un servidor RaptorXML Server, puede ordenar al servidor que escriba el resultado de la asignación en el archivo que se pasa como valor del parámetro `--output`. Para escribir el resultado en un archivo, añada o edite el parámetro `--output` en el archivo **DoTransform.bat**. Por ejemplo, el archivo **DoTransform.bat** que aparece en el comando siguiente se editó para que se escriba el resultado de la asignación en el archivo **Resultado.txt** (véase el texto resaltado).

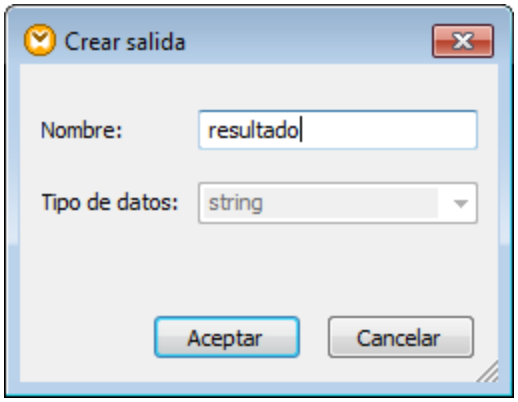
	`RaptorXML xslt --xslt-version=2 --input="MappingMapToResult1.xslt" --output="Resultado.txt" %* "MappingMapToResult1.xslt"` Si no se definió el parámetro `--output`, el resultado de la asignación se escribirá en la secuencia de datos de salida estándar (stdout) cuando se ejecute la asignación.

Cuando cree una asignación invertida (con el comando de menú **Herramientas | Crear asignación inversa**), los componentes de salida simples se convertirán en componentes de entrada simples.

5.2.1 Agregar componentes de salida simples

Para agregar un componente de salida en el área de asignación:

1. Asegúrese de que en la ventana de asignación está activa la asignación principal (y no una función definida por el usuario).
2. En el menú **Función** haga clic en el comando **Insertar componente de salida**.
3. Introduzca el nombre del componente.
4. Haga clic en **Aceptar**.



Cuadro de diálogo "Crear salida"

El nombre del componente se puede cambiar más adelante de varias maneras:

- Seleccionando el componente y haciendo clic en **Componente | Propiedades**.
- Haciendo doble clic en el título del componente.
- Haga clic con el botón derecho en el título del componente y después elija **Propiedades**.

5.2.2 Ejemplo: vista previa de resultados de una función

Este ejemplo sirve para explicar cómo consultar la vista previa de los resultados que devuelven funciones de MapForce con la ayuda de componente de salida simples. Antes de consultar este ejemplo es conveniente

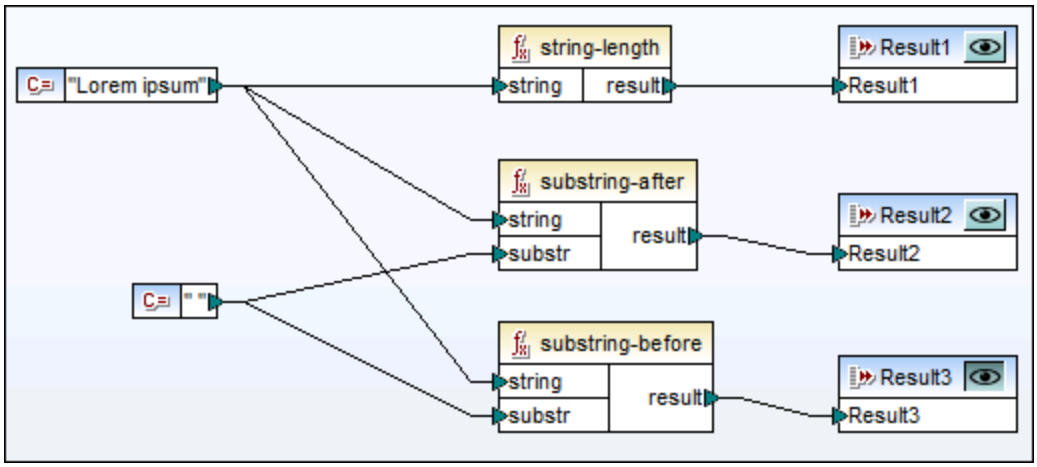
conocer el funcionamiento básico de las funciones en general y de las funciones de MapForce en particular. Puede encontrar más información en la sección Funciones[196].

Nuestro objetivo en este ejemplo es añadir varias funciones al área de asignación y aprender a generar la vista previa de los resultados con ayuda de componentes de salida simples. En concreto, este ejemplo utiliza varias funciones simples de la biblioteca principal **core** de MapForce. Aquí resumimos el uso de estas funciones:

string-length[313]	Devuelve el número de caracteres de la cadena que se dio como argumento. Por ejemplo, si pasamos a esta función el valor "Lorem ipsum", se obtiene el resultado "11" porque es el número de caracteres que toma el texto "Lorem ipsum".
substring-after[314]	Devuelve la parte de la cadena que aparece después del separador que se dio como argumento. Por ejemplo, si pasamos a esta función el valor "Lorem ipsum" y el carácter de espaciado (" "), el resultado es "ipsum".
substring-before[315]	Devuelve la parte de la cadena que aparece antes del separador que se dio como argumento. Por ejemplo, si pasamos a esta función el valor "Lorem ipsum" y el carácter de espaciado (" "), el resultado es "Lorem".

Siga estos pasos para probar todas estas funciones con un valor de texto (p. ej. "Lorem ipsum"):

1. Añada una constante con el valor "Lorem ipsum" al área de asignación (con el comando de menú **Insertar | Constante**). La constante será el parámetro de entrada para cada una de las funciones que vamos a probar.
2. Añada las funciones **string-length**, **substring-after** y **substring-before** al área de asignación (arrastrándolas desde la biblioteca **core**, sección **string functions**, hasta el área de asignación).
3. Añada una constante con un carácter de espaciado como valor (" "). Este será el parámetro separador que requieren las funciones **substring-after** y **substring-before**.
4. Añada tres componentes de salida simples (con el comando de menú **Función | Insertar componente de salida**). En nuestro ejemplo estos componentes se llaman **Resultado1**, **Resultado2** y **Resultado3** respectivamente.
5. Conecte los componentes tal y como aparece en la siguiente imagen.



Probando resultados de funciones con componentes de salida simples

Tal y como puede ver en la imagen anterior, la cadena "Lorem ipsum" hace de parámetro de entrada para las funciones **string-length**, **substring-after** y **substring-before**. Además, las funciones **substring-after**

y **`substring-before`** toman como segundo parámetro de entrada un valor de espaciado. Los componentes **Resultado1**, **Resultado2** y **Resultado3** pueden utilizarse para generar la vista previa del resultado de cada función.

Para obtener la vista previa de resultados de una función:

- Haga clic en el botón **Vista previa** () en la barra de título del componente correspondiente y después haga clic en el panel *Resultados* de la ventana de asignación.

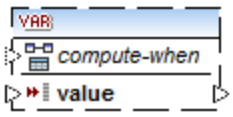
5.3 Variables

Las variables son un tipo especial de componente utilizado para almacenar un resultado de asignación intermedio y poder seguir procesándolo. Las variables pueden ser de tipo simple (p.ej. cadena, entero, valor booleano, etc) o de tipo complejo (estructuras jerárquicas).

Una de las cosas más importantes de las variables es que son secuencias y que pueden utilizarse para crear secuencias. En este sentido el término secuencia significa "una lista de cero o más elementos" (*véase* *Reglas y estrategias de asignación de datos* 407). Esto permite a las variables poder procesar varios elementos durante todo el ciclo de vida de la asignación. No obstante, también puede asignar un valor a una variable una sola vez y conservarlo para el resto de la asignación (*véase* *Cambiar el contexto y ámbito de las variables* 166).

Variables simples

Las variables simples se generan para representar tipos atómicos como cadenas de texto, números y elementos booleanos (*imagen siguiente*).



Variables complejas

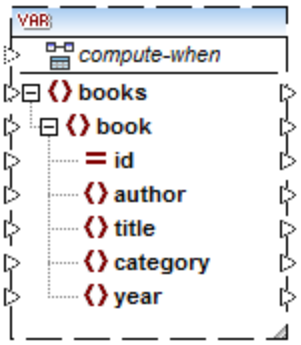
Las variables complejas tienen estructura en forma de árbol. Estas son las estructuras en que pueden basarse las variables complejas.

MapForce Basic Edition:
- Estructura XML Schema

MapForce Professional Edition:
- Estructura XML Schema
- Estructura de BD

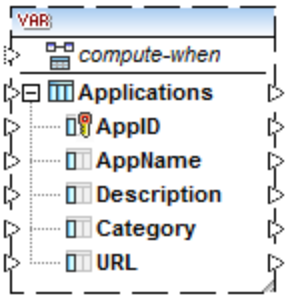
MapForce Enterprise Edition:
- Estructura XML Schema
- Estructura de BD
- Estructura EDI
- Estructura FlexText
- Estructura JSON Schema

Ejemplo 1: variable basada en un esquema XML
Puede crear una variable de tipo complejo suministrando un esquema XML que exprese la estructura de la variable. Si el esquema define elementos globalmente, podrá elegir cuál de ellos debe ser el nodo raíz de la estructura de la variable. Recuerde que una variable no tiene ningún archivo XML de instancia asociado sino que sus datos se calculan cuando se ejecuta la asignación.



Ejemplo 2: variable basada en una BD

Este ejemplo (*imagen siguiente*) sólo es relevante para las ediciones MapForce Professional y Enterprise. Si elige una estructura de BD para la variable, puede elegir una tabla concreta de la base de datos como elemento raíz de la estructura de la variable.

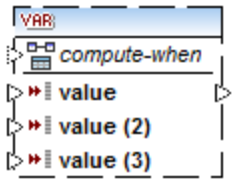


Compute-when

En las imágenes anteriores observará que cada variable tiene un elemento llamado `compute-when`. Conectar este elemento es opcional y sirve para controlar cómo se debe calcular el valor de la variable en la asignación (*véase* *Cambiar el contexto y el ámbito de las variables* 166).

Variables simples con entradas duplicadas

Si es necesario, los elementos de la estructura de una variable se pueden duplicar para que acepten datos de más de una conexión de origen, igual que se hace con componentes estándar (*véase* *Duplicar entradas* 40). Sin embargo, esto no se puede hacer en las variables creadas a partir de tablas de BD. En la imagen siguiente puede ver un ejemplo de variable simple con entradas duplicadas.



Variables encadenadas vs. variables

Hasta cierto punto las variables pueden compararse con los componentes intermedios de una asignación en cadena (*véase* *Asignación encadenada* 94). Sin embargo, son mucho más flexibles y prácticas cuando no se necesita generar archivos intermedios en cada fase de la asignación. A continuación se explican las diferencias que existen entre las variables y las asignaciones en cadena.

Asignaciones en cadena	**Variables**
Las asignaciones en cadena se desarrollan en dos fases totalmente independientes. Por ejemplo, imaginemos que una asignación tiene tres componentes llamados A, B y C. La ejecución de la asignación se desarrolla en dos fases: la ejecución de la asignación desde A hasta B y la ejecución de la asignación desde B hasta C.	Mientras se ejecuta la asignación, las variables se evalúan en función de su contexto y su ámbito. Su contexto y ámbito puede modificarse (*véase* *Cambiar el contexto y ámbito de las variables* 166).
Cuando se ejecuta la asignación, los resultados intermedios se almacenan de forma externa en archivos.	Cuando se ejecuta la asignación, los resultados intermedios se almacenan de forma internan. No se producen archivos externos con los resultados de la variable.
Con el botón de vista previa se puede consultar la vista previa del resultado intermedio.	No se puede consultar la vista previa del resultado de una variable porque éste se calcula en tiempo de ejecución.

Nota: No se admite el uso de variables si el lenguaje de transformación elegido es XSLT 1.0.

5.3.1 Agregar variables

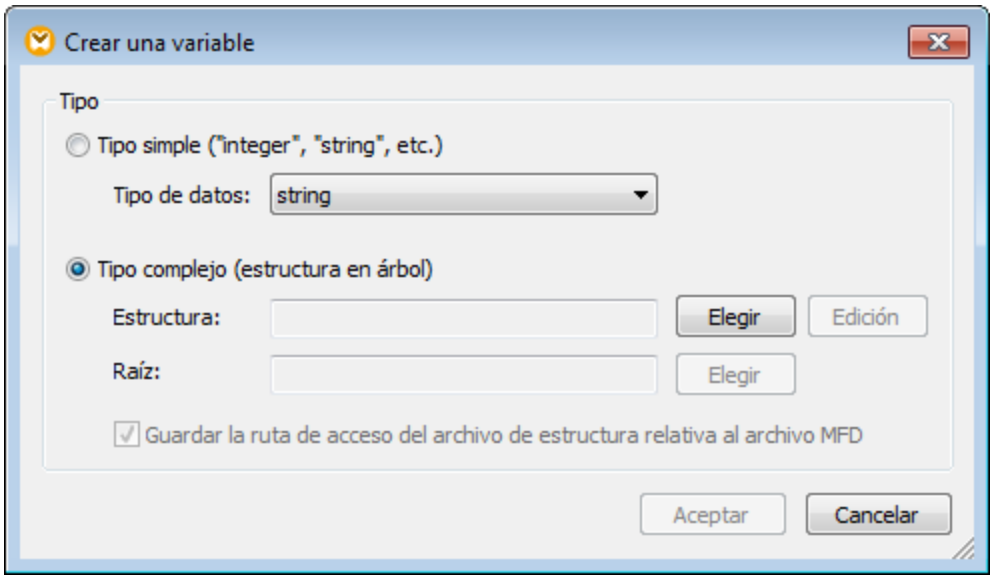
En este apartado explicamos cómo agregar variables a una asignación. La primera opción es agregar variables con comandos de menú o de la barra de herramientas. La segunda opción permite agregar variables desde el menú contextual.

Opción 1: con comandos de menú o de la barra de herramientas

Esta opción sirve para agregar variables con comandos de menú o de la barra de herramientas. Siga estos pasos:

1. En el menú **Insertar** haga clic en el comando **Variable**. También puede hacer clic en el botón 

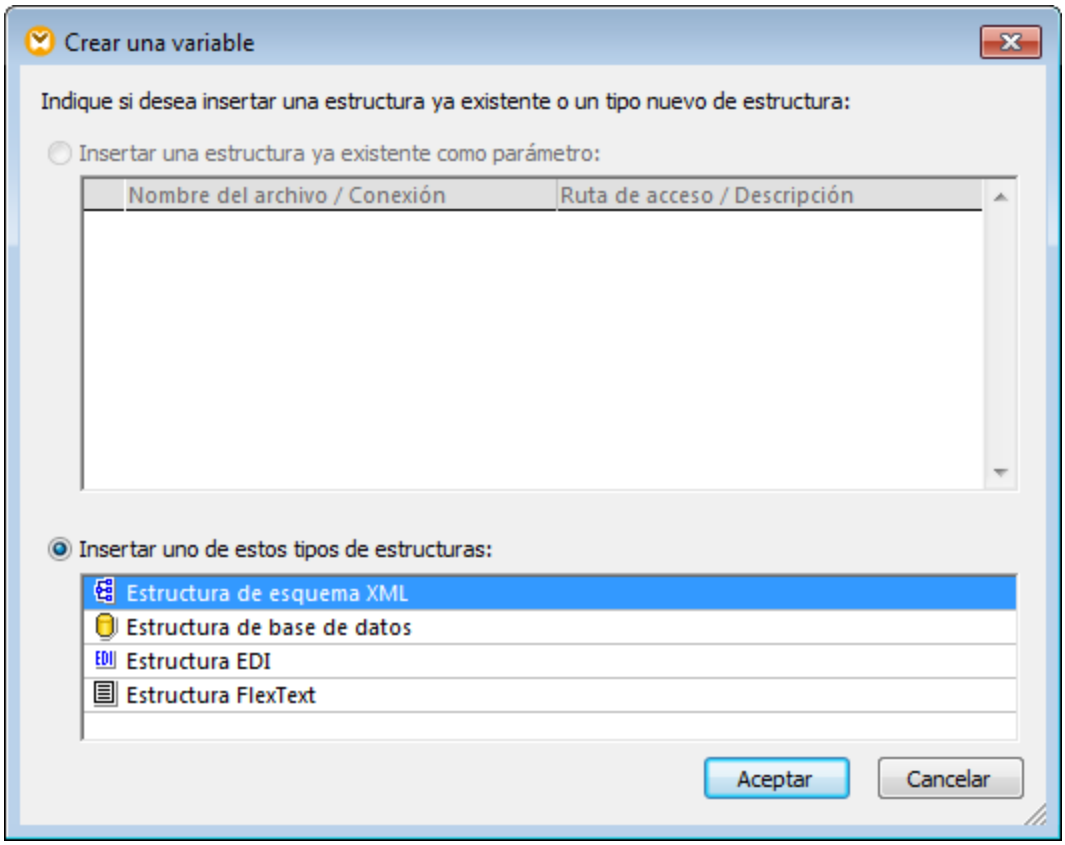
(**Variable**) de la barra de herramientas.



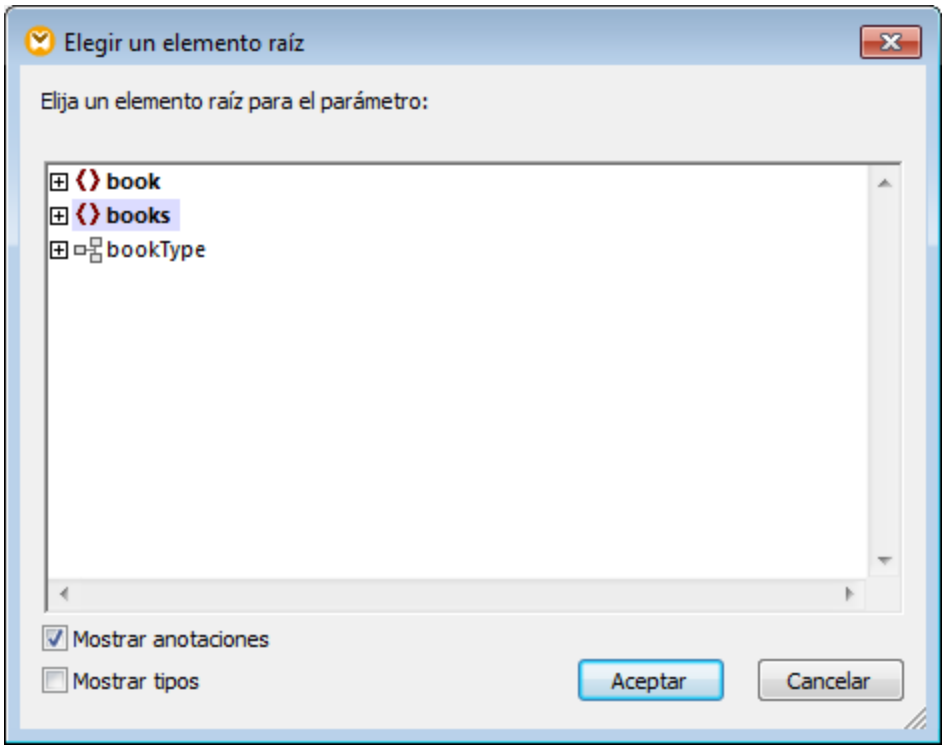
2. Seleccione el tipo de variable que quiere insertar (de tipo simple o complejo).

Si selecciona **Tipo complejo** debe seguir también estos pasos:

3. Haga clic en **Elegir** para seleccionar el origen del que se debe tomar la estructura de la variable[160]. Las estructuras que ve en la imagen siguiente se aplican solamente a MapForce Enterprise Edition. Consulte la lista de estructuras relevantes para las otras ediciones de MapForce en el apartado anterior[160].



4. Cuando la aplicación se lo pida indique el elemento raíz de la estructura de la variable. Por ejemplo, en los esquemas XML puede seleccionar cualquier elemento o tipo de la fuente seleccionada (*imagen siguiente*).

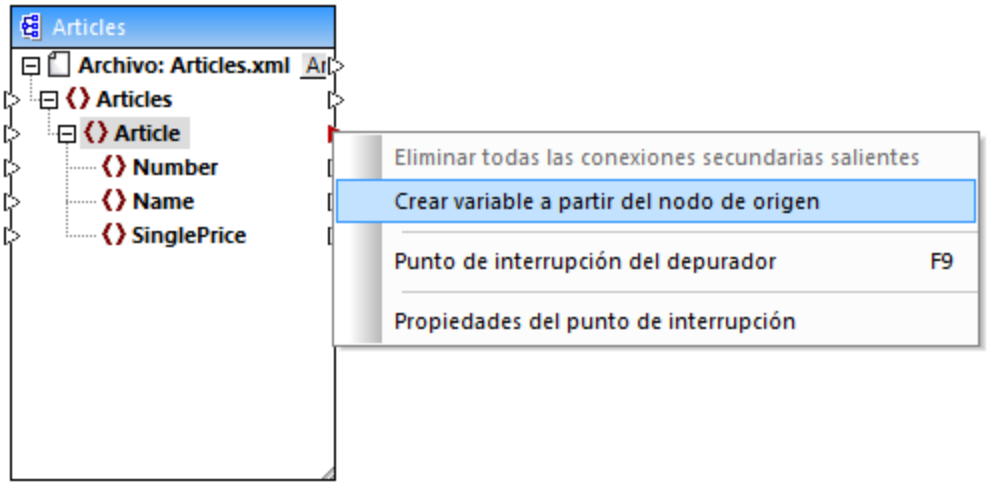


Opción 2: desde el menú contextual

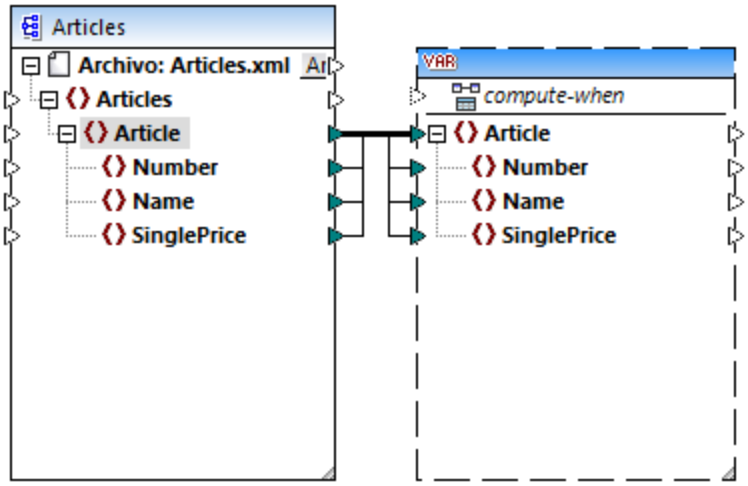
La segunda opción permite crear variables desde el menú contextual. A continuación se enumeran todas las opciones posibles.

Variable a partir de un nodo de origen

Para crear una variable a partir de un nodo de origen, haga clic con el botón derecho en el conector de salida de un componente (en este ejemplo usamos el del elemento `<Article>`) y seleccione **Crear variable a partir del nodo de origen** (*imagen siguiente*).



Se crea una variable compleja con el esquema de origen del componente `Articles`. Todos los elementos se conectan automáticamente con una conexión de copia total[55] (*imagen siguiente*).

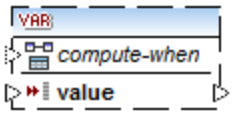


Variable a partir de un nodo de destino
Para crear una variable a partir de un nodo de destino, haga clic con el botón derecho en el conector de entrada de un componente de destino y seleccione **Crear variable a partir del nodo de destino**. Se crea una variable compleja con el mismo esquema del componente de destino. Todos los elementos se conectan automáticamente con una conexión de copia total.

Variable a partir de un filtro:
Para crear una variable usando un filtro, haga clic con el botón derecho en el conector de salida de un componente de filtro (`on-true/on-false`) y seleccione **Crear variable a partir del nodo de origen**. Se crea un componente complejo con el esquema de origen y usa automáticamente el elemento vinculado al filtro de entrada como elemento raíz del componente intermedio.

5.3.2 Contexto y ámbito de las variables

Todas las variables tienen un elemento de entrada `compute-when` que sirve para controlar el ámbito de la variable, es decir, en qué momento y con qué frecuencia se calcula el valor de la variable cuando se ejecuta la asignación. En muchos casos este elemento de entrada puede dejarse sin conectar, pero en ocasiones puede ser imprescindible a la hora de reemplazar el contexto predeterminado o para optimizar el rendimiento de la asignación de datos.



El término ***subestructura*** utilizado en esta documentación hace referencia al conjunto de elementos/nodos de un componente de destino y a todos sus descendientes (p. ej. un elemento `<Persona>` con sus elementos secundarios `<Nombre>` y `<Apellido>`.

El término ***valor de variable*** sirve para denominar los datos disponibles en el lado saliente del componente de variable.

- En el caso de las variables simples se trata de una secuencia de valores atómicos cuyo tipo de datos está especificado en las propiedades del componente.+
- En el caso de las variables complejas se trata de una secuencia de nodos raíz (del tipo especificado en las propiedades del componente), cada uno de ellos con sus nodos descendientes.

La secuencia de valores atómicos (o nodos) puede contener un elemento e incluso cero. Esto depende de cómo esté conectado el lado entrante de la variable y de con qué elementos principales del componente de origen/destino esté conectado.

Si Compute-when no está conectado (configuración predeterminada)

Si el elemento de entrada `compute-when` no está conectado (a ningún nodo de salida de un componente de origen), entonces el valor de la variable se calcula *cuando se use por primera vez en una subestructura de destino* (ya sea por conexión directa entre el componente de variable y un nodo del componente de destino o por conexión indirecta a través de funciones). El mismo valor de variable se utiliza después para todos los nodos secundarios de destino incluidos dentro de la subestructura.

El valor real de la variable dependerá de las conexiones que existan entre los elementos principales del componente de origen y de destino. Este comportamiento predeterminado es idéntico al de los resultados complejos de funciones definidas por el usuario[208] y de llamadas a función de servicio web. Si el resultado de la variable se conecta a varios nodos de destino que no guardan relación entre sí, el valor de la variable se calcula por separado para cada uno de ellos. Esto puede dar lugar a resultados distintos porque las distintas conexiones principales influyen en el contexto en el que se evalúa el valor de la variable.

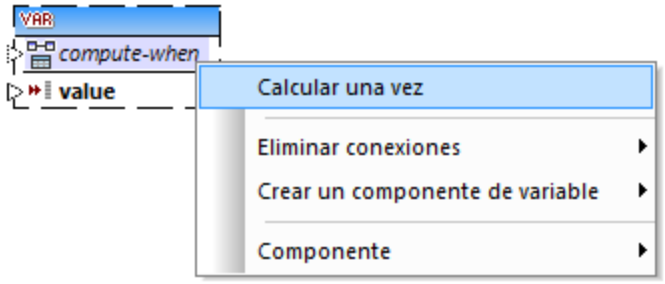
Si Compute-when está conectado

Si se conecta un conector de salida de un componente de origen con el elemento `compute-when`, el valor de la variable se calcula *cuando dicho elemento de origen se use por primera vez en una subestructura de destino*.

En realidad la variable hace las veces de elemento secundario del elemento que está conectado a `compute-when` (*imagen siguiente*). Esto permite vincular la variable con un elemento de origen concreto. Es decir, en tiempo de ejecución la variable se vuelve a evaluar cada vez que se lea un elemento nuevo de la secuencia del componente de origen. Se trata de la regla general que rige las conexiones en MapForce: *por cada elemento de origen, se crea uno de destino*. En el caso del elemento de entrada `compute-when` esto significa que *por cada elemento de origen, se calcula el valor de variable* (*véase* *Reglas y estrategias de asignación de datos*[407]).

Compute-once

Si fuera necesario, puede solicitar que el valor de variable se calcule una sola vez antes de cada componente de destino, lo cual convierte a la variable en una constante global para el resto de la asignación. Esto se consigue haciendo clic con el botón derecho en el elemento `compute-when` y seleccionando el comando **Calcular una vez** en el menú contextual (*imagen siguiente*).

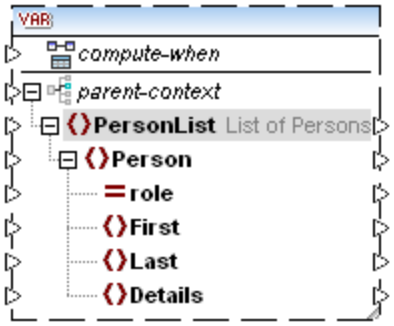


Cuando se cambia el ámbito de una variable por `compute-when=once`, el conector de entrada se elimina del elemento `compute-when` (porque dicha variable solamente se evaluará una vez). En funciones definidas por el usuario la variable `compute-when=once` se evalúa cada vez que se llama a la función y antes de que el resultado de la función se evalúe.

Contexto primario

En ocasiones puede ser necesario agregar un contexto primario. Por ejemplo, si la asignación usa varios filtros y necesita recorrer un nodo primario adicional (*véase* *Reemplazar el contexto de la asignación* 415).

Para agregar un contexto primario a una variable basta con hacer clic con el botón derecho en el nodo raíz (p. ej. "PersonList") y seleccionar el comando b en el menú contextual. Como resultado se añade un nodo nuevo llamado `parent-context` a la jerarquía de la variable.

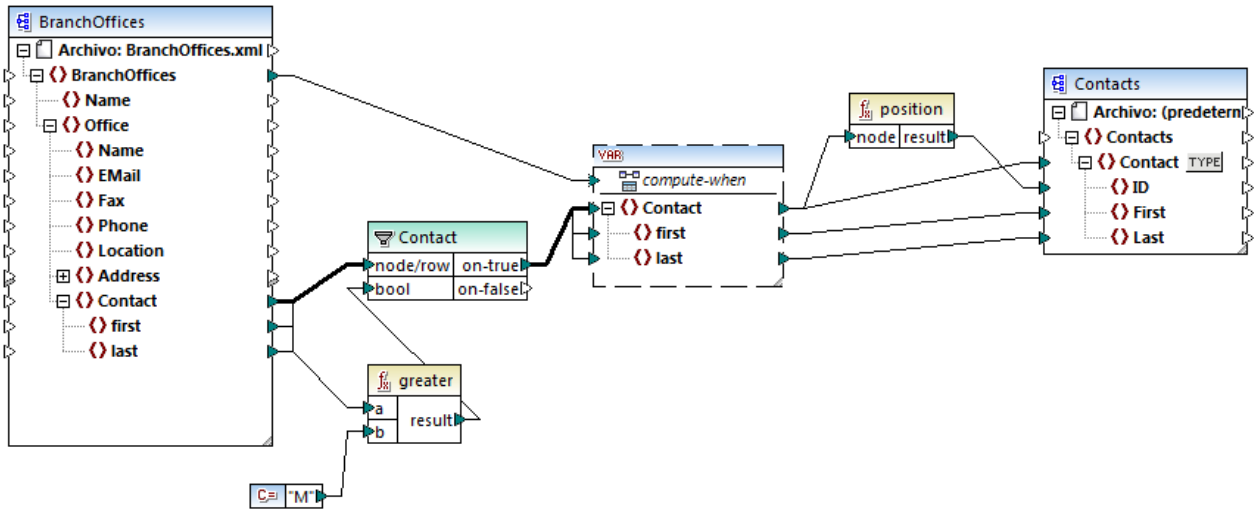


El contexto primario añade un nodo primario virtual a la jerarquía del componente, lo cual permite recorrer otro nodo más en el mismo componente de origen o en otro distinto.

5.3.3 Ejemplo: filtrar y numerar nodos

El diseño de asignación que aparece más abajo corresponde al archivo **PositionInFilteredSequence.mfd** situado en la carpeta **<Documentos>\Altova\MapForce2024\MapForceExamples**.

Esta asignación lee un archivo XML que contiene datos de contacto de varias persona, los filtra y los escribe en un archivo XML de destino. El objetivo del diseño es filtrar datos del archivo XML de origen de las personas cuyo apellido empiece a partir de la letra M. Además, los contactos extraídos deben estar numerados en el XML de destino. Este número hará de identificador único en el XML de destino.



PositionInFilteredSequence.mfd

Para conseguir este objetivo se añadieron varios componentes al diseño de asignación:

- Un filtro (*véase* *Filtros y condiciones* 178)
- Una variable compleja (*véase* *Agregar variables* 162)
- Las funciones **`greater`** 263 y **`position`** 300 (*véase* *Agregar una función* 197)
- Una constante (para agregar una constante seleccione el comando de menú **Insertar | Constante**).

La variable usa el mismo esquema que el componente de origen. Si hacemos clic con el botón derecho en la variable y seleccionamos Propiedades en el menú contextual, veremos que el nodo raíz que está seleccionado para esta estructura de variable es **BranchOffices/Office/Contact**.

Primero se pasan al filtro los datos del componente de origen. Después el filtro pasa a la variable los registros que cumplan la condición de filtrado. Es decir, el filtro está configurado para obtener los nodos `Contact` cuyo nombre sea igual o mayor que "M". Para ello se usa la función **`greater`** 263 que compara cada elemento `last` con el valor de constante "M".

La variable tiene la entrada `compute-when` conectada con el elemento raíz del componente de origen (`BranchOffices`). En tiempo de ejecución esto hace que la variable se vuelva a evaluar cada vez que se lea un elemento nuevo de la secuencia del componente de origen. Sin embargo, en esta asignación no influye en modo alguno el conectar o no conectar el elemento `compute-when`. El motivo es que la variable está conectada al elemento de origen `Contact` (de forma indirecta a través del filtro) y se calcularía tantas veces como instancias de `Contact` cumplan la condición de filtrado.

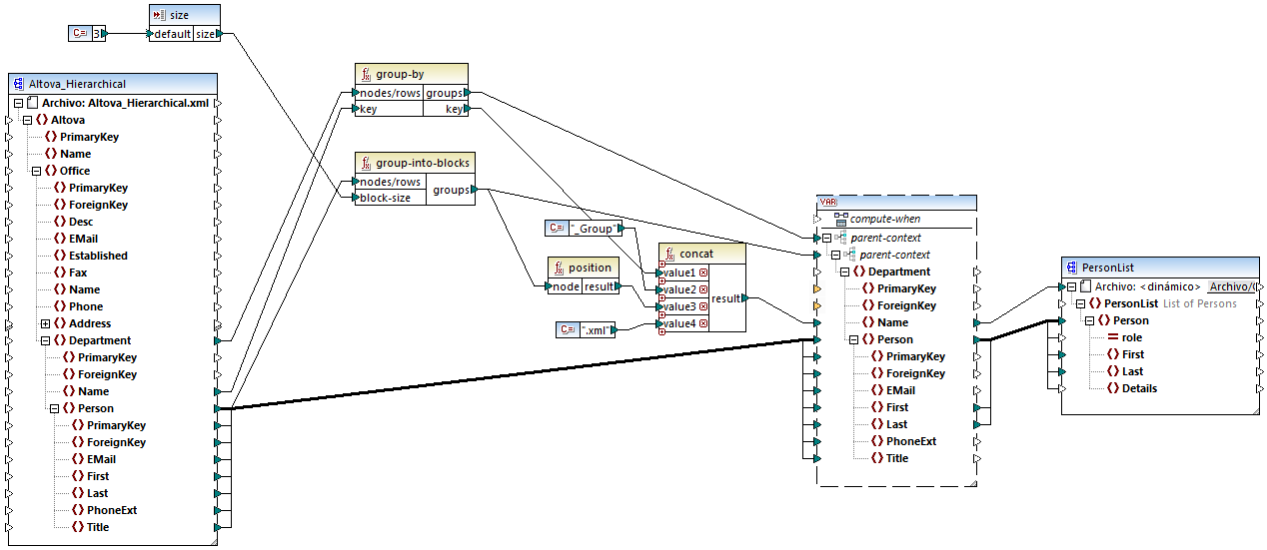
La función **`position`** 300 devuelve (por cada iteración de la variable) el número de la secuencia actual. Sólo ocho contactos cumplen la condición. Por tanto, si consultamos la vista previa del resultado de la asignación, veremos que se escribieron los identificadores 1 - 8 en el elemento `ID` del componente de destino.

La variable se necesita porque necesitamos numerar todos los registros. Si se hubiera conectado el resultado del filtro al componente de destino directamente, no habría manera de numerar cada instancia de `Contact`. La función de la variable es, por tanto, almacenar temporalmente cada instancia de `Contact` para poder numerarlas antes de escribirlas en el destino.

5.3.4 Ejemplo: crear grupos y subgrupos de registros

El diseño de asignación que aparece más abajo corresponde al archivo **DividePersonsByDepartmentIntoGroups.mfd** situado en la carpeta **<Documentos>\Altova\MapForce2024\MapForceExamples**.

Este diseño procesa un archivo XML que contiene registros de empleados de una compañía ficticia. La compañía tiene dos oficinas: "Nanonull, Inc." y "Nanonull Partners, Inc". Cada oficina tiene varios departamentos (p. ej. "IT", "Marketing", etc.) y cada departamento tiene como mínimo un empleado. El objetivo de esta asignación de datos es crear grupos formados por un tres personas como máximo de cada departamento, sin importar la oficina. El tamaño predeterminado de cada grupo es 3, pero esto se puede cambiar. Además cada grupo debe guardarse en un archivo XML distinto, cuyo nombre seguirá el patrón "<Nombre Departamento>_NºGrupo" (p. ej. **Marketing_Group1.xml**, **Marketing_Group2.xml**, etc.).



DividePersonsByDepartmentIntoGroups.mfd

Como puede ver, la asignación cuenta con una variable compleja y varios componentes más (funciones sobre todo). La variable tiene la misma estructura que el elemento `Department` del archivo XML de origen. Si hacemos clic con el botón derecho en la variable para ver sus propiedades, observaremos que utiliza el mismo esquema que el componente de origen y que su elemento raíz es `Department`. Y lo que es más importante, la variable tiene dos elementos `parent-context` anidados, que garantizan que la variable se calcule primero en el contexto de cada departamento y después en el contexto de cada grupo dentro de cada departamento (*véase* *Cambiar el contexto y ámbito de las variables* 166).

En un principio la asignación recorre todos los departamentos para obtener el nombre de cada uno de ellos (el nombre de los departamentos se necesita para crear el nombre de archivo que corresponde a cada grupo). Esto se consigue conectando la función **`group-by`** 287 al elemento de origen `Department` y pasando el nombre del departamento como clave de agrupación.

Después, dentro del contexto de cada departamento, se lleva a cabo otra agrupación: la asignación llama a la función **`group-into-blocks`** 292 para crear los grupos necesarios de empleados. El tamaño de cada grupo viene dado por un componente de entrada simple cuyo valor predeterminado es "3". El valor predeterminado

viene dado por una constante. En este ejemplo, para cambiar el tamaño de cada grupo basta con modificar el valor de la constante según corresponda. Sin embargo, también se puede modificar el componente de entrada **size** para que, si la asignación se ejecuta con código generado o con MapForce Server, el tamaño de cada grupo pueda especificarse como parámetro (*véase* *Pasar parámetros a la asignación* 147).

A continuación, el valor de la variable se pasa al componente XML de destino PersonList. El nombre de archivo de cada grupo creado se calcula con ayuda de la función **concat** 310, mediante la concatenación de:

1. el nombre de cada departamento,
2. la cadena "_Group",
3. el número que tiene el grupo en la secuencia actual (p. ej. "1" si se trata del primer grupo del departamento) y
4. la cadena ".xml"

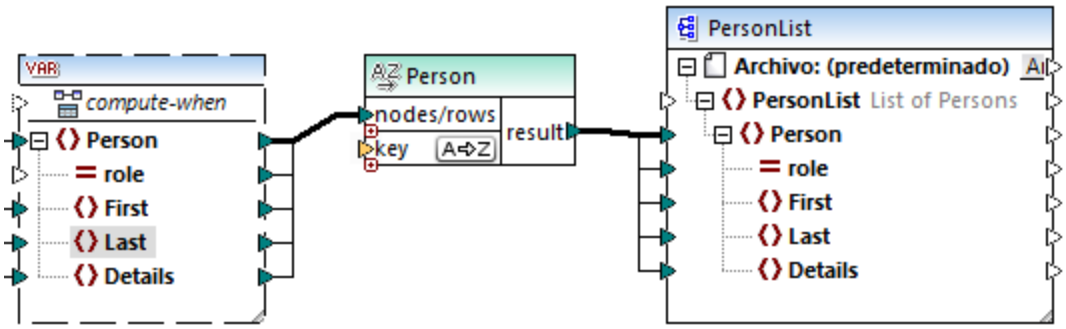
El resultado de la concatenación se almacena en el elemento `Name` de la variable y después se pasa como nombre de archivo dinámico al componente de destino. Por tanto, se crea un nombre de archivo nuevo por cada valor recibido. En este ejemplo, la variable calcula 8 grupos en total, así que se crean 8 archivos de salida cuando se ejecuta la asignación. Para más información consulte la sección Procesar varios archivos de entrada o salida simultáneamente 402.

5.4 Ordenar componentes

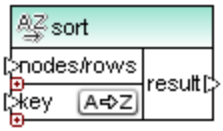
Para ordenar datos de entrada siguiendo un criterio de ordenación concreto basta con usar un componente de ordenación. El componente de ordenación es compatible con estos lenguajes de destino: XSLT2, XQuery y el motor de ejecución integrado.

Para agregar un componente de ordenación a la asignación:

- Haga clic con el botón derecho en una conexión y seleccione **Insertar componente de ordenación: nodos/filas** en el menú contextual. Esto inserta un componente de ordenación y lo conecta automáticamente a los componentes de origen y destino. Por ejemplo, en la imagen siguiente el componente de ordenación se introdujo entre una variable y un componente XML. Lo único que falta por conectar a mano es el criterio de ordenación (el campo por el que desea ordenar los elementos).



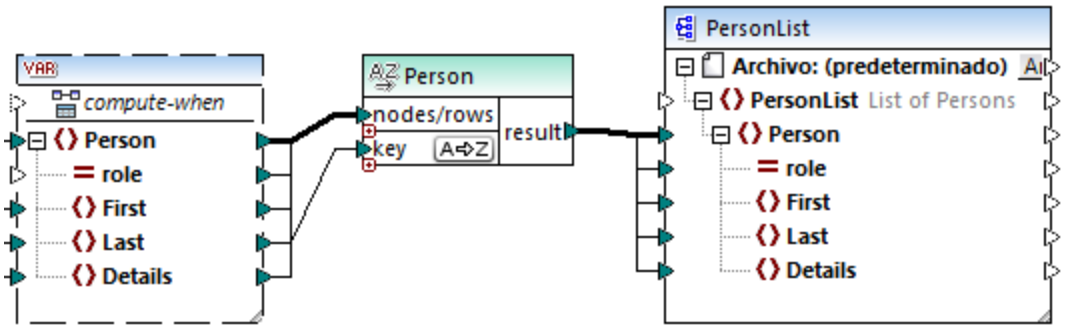
- En el menú **Insertar** seleccione el comando **Componente de ordenación: nodos/filas** (o haga clic en el botón **Insertar componente de ordenación** de la barra de herramientas). Esto inserta el componente de ordenación pero sin conectarlo.



En cuanto conectemos el componente de ordenación con el componente de origen, en la barra de título del componente de ordenación aparecerá el nombre del elemento que está conectado a `nodes/rows`.

Para definir por qué elemento se deben ordenar los nodos/filas:

- Conecte el elemento por el que desea ordenar los nodos/filas al parámetro `key` del componente de ordenación. Por ejemplo, en la imagen siguiente los nodos/filas `Person` se ordenan por el campo `Last`.

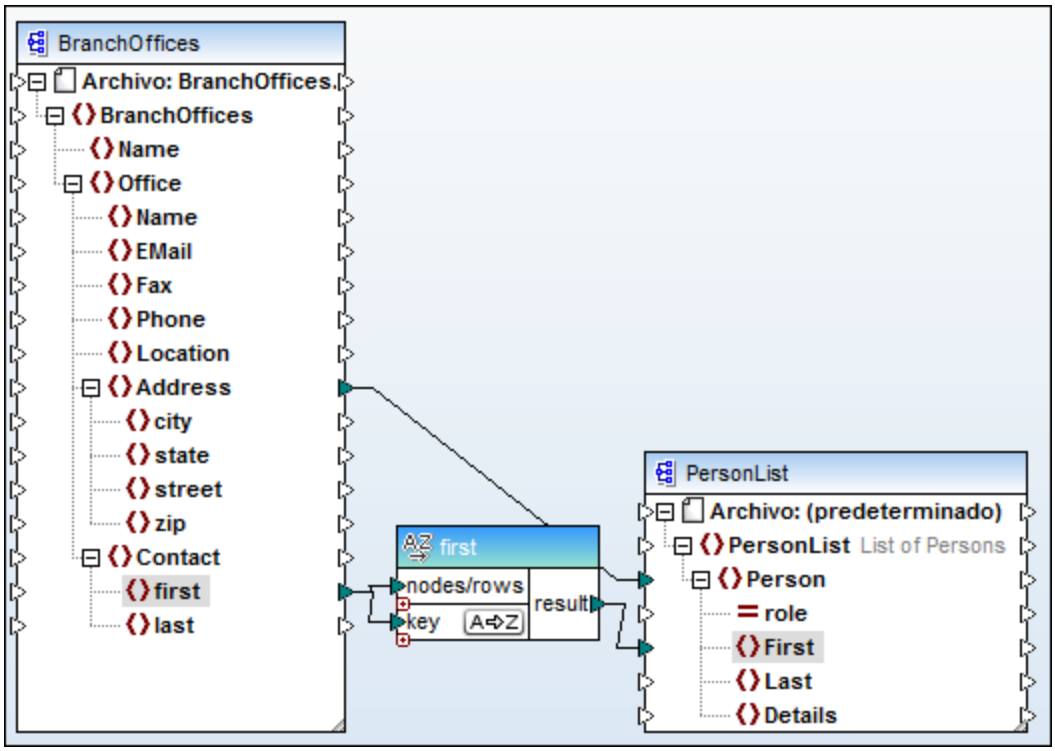


Para cambiar el orden:

- Haga clic en el icono A⇨Z del componente de ordenación. Se convertirá en el icono Z➔A para mostrar que el orden se invirtió.

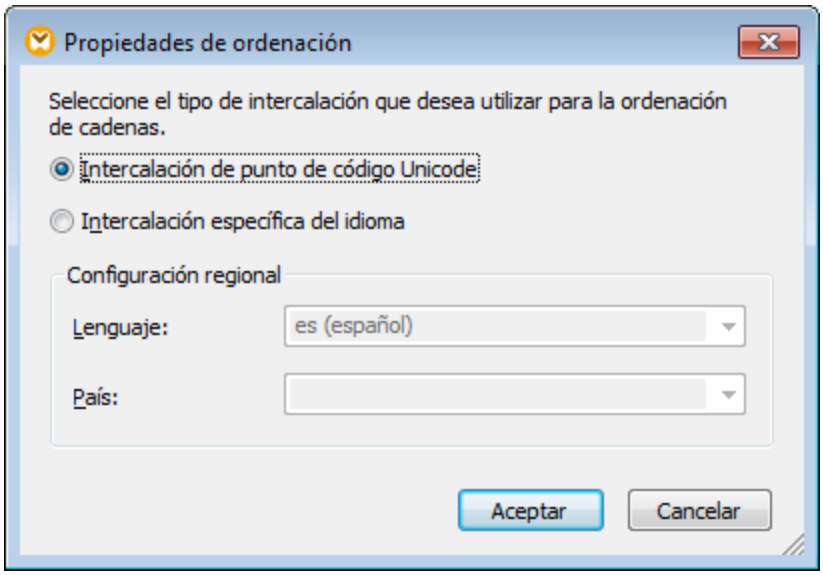
Para ordenar datos de entrada compuestos por elementos de tipo simple:

- Conecte el elemento tanto al parámetro `nodes/rows` como al parámetro `key` del componente de ordenación. En la imagen siguiente, por ejemplo, se ordena el elemento de tipo simple `first`.



Para ordenar cadenas usando reglas de un idioma concreto:

- Haga doble clic en el título del componente de ordenación para abrir el cuadro de diálogo "Propiedades de ordenación" (*imagen siguiente*).

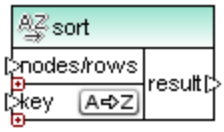


Intercalación de punto de código: esta opción predeterminada compara/ordena las cadenas basándose en valores de punto de código. Los valores de punto de código son enteros que se asignaron a caracteres abstractos del conjunto de caracteres universal adoptado por el consorcio Unicode. Esta opción permite ordenar datos en muchos idiomas y scripts.

Intercalación específica del idioma: esta opción permite definir el idioma y el país que debe utilizarse para la ordenación. Esta opción es compatible con el motor de ejecución integrado. Para XSLT la compatibilidad dependerá del motor que se utilice para ejecutar el código.

5.4.1 Ordenar según varias claves

Cuando se añade un componente de ordenación a la asignación, MapForce crea por defecto un criterio de ordenación llamado key.



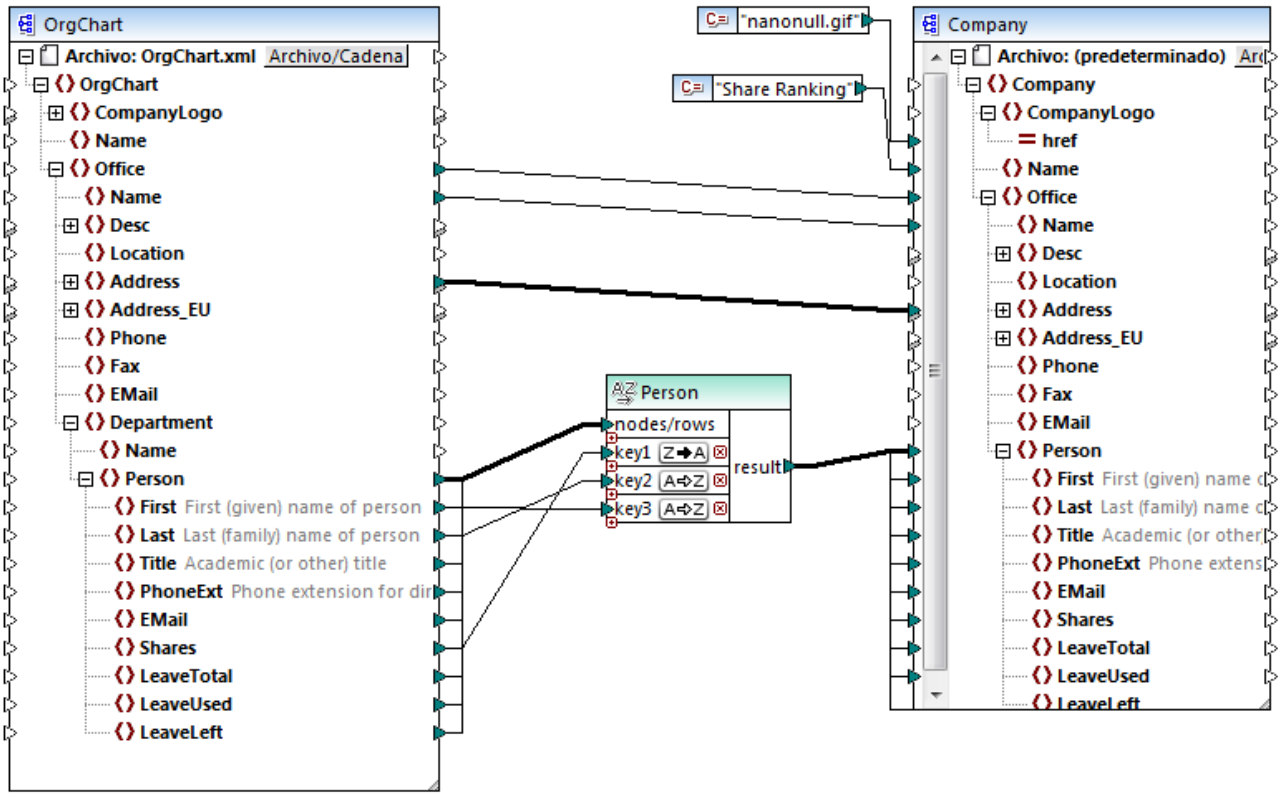
Componente de ordenación

Si desea usar varios criterios de ordenación, deberá realizar varios ajustes en el componente:

- Haga clic en el icono **Agregar criterio** (⊡) para agregar un criterio nuevo (p. ej. key2).
- Haga clic en el icono **Eliminar criterio** (⊠) para eliminar un criterio de ordenación.

- Arrastre y coloque una conexión encima del icono para agregar un criterio y al mismo tiempo conectarlo.

El diseño de asignación **<Documentos>\Altova\MapForce2024\MapForceExamples\SortByMultipleKeys.mfd** contiene un componente de ordenación que usa varios criterios.



SortByMultipleKeys.mfd

En esta asignación de datos los registros `Person` se ordenan siguiendo tres criterios:

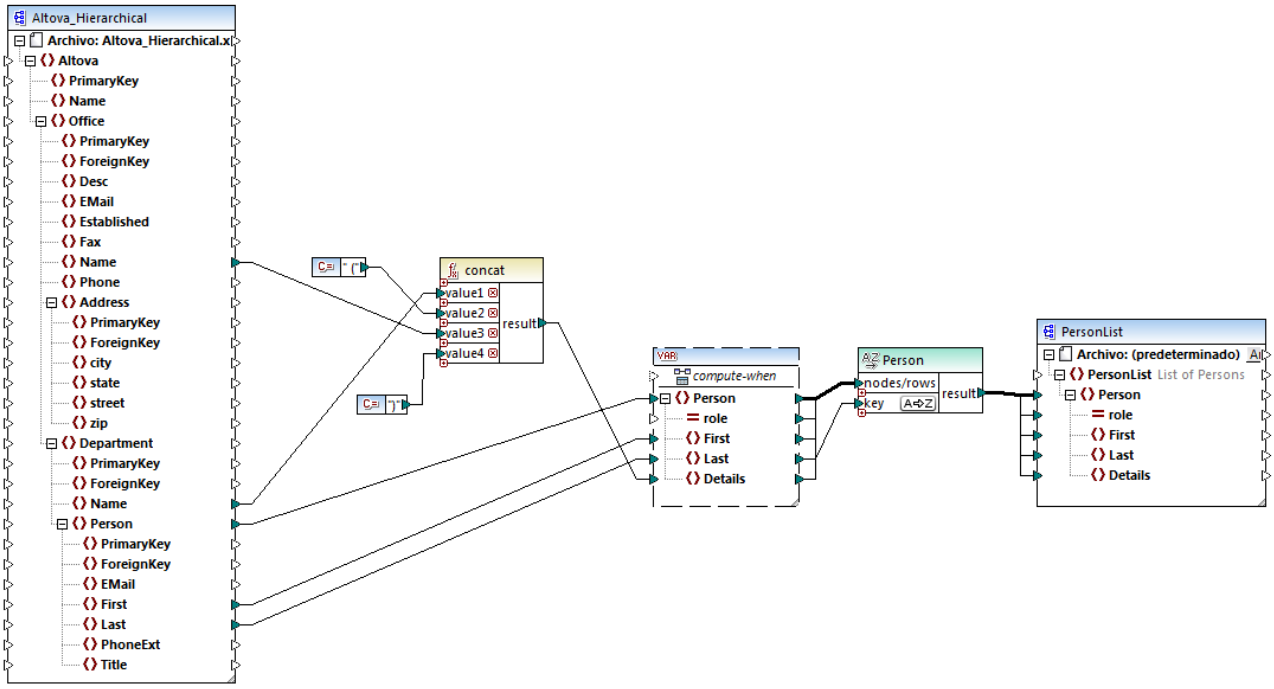
1. `Shares` (número de acciones en propiedad de cada persona)
2. `Last` (apellido)
3. `First` (nombre)

Observe que la posición del criterio de ordenación en el componente determina su prioridad. Por ejemplo, en la imagen puede ver que los registros se orden primero en función del número de acciones. Se trata del criterio de ordenación con mayor prioridad. Si el número de acciones fuera el mismo, las personas se ordenan por apellido. Y, por último, si varios personas tuvieran el mismo apellido, se ordenarán por nombre.

El orden de cada criterio puede ser distinto. Por ejemplo, en la imagen anterior puede ver que el criterio `Shares` indica un orden descendiente (Z-A), mientras que los otros dos criterios siguen un orden ascendente (A-Z).

5.4.2 Ordenar con variables

En algunos casos puede ser necesario agregar variables intermedias para conseguir ciertos resultados. En el diseño de asignación que aparece a continuación se extraen registros de un archivo XML y éstos se orden con ayuda de variables intermedias. Se trata del diseño de asignación **<Documentos>\Altova\MapForce2024\MapForceExamples\Altova_Hierarchical_Sort.mfd**.



Altova_Hierarchical_Sort.mfd

Esta asignación lee datos de un archivo XML de origen llamado **Altova_Hierarchical.xml** y los escribe en un archivo XML de destino. Como puede ver, el XML de origen contiene información sobre una compañía ficticia. La compañía se divide en oficinas y éstas en departamentos. A su vez, los departamentos están compuestos por empleados.

El componente XML de destino `PersonList` contiene una lista de registros `Person`. El elemento `Details` tiene la función de almacenar información sobre la oficina y el departamento al que pertenece cada empleado.

El objetivo de esta asignación es extraer todos los empleados del XML de origen y ordenarlos por apellido y por orden alfabético. Además, el nombre de la oficina y del departamento al que pertenece cada empleado debe escribirse en el elemento `Details`.

Estos son los componentes necesarios para conseguir el objetivo de la asignación:

1. La función **`concat`**, que devuelve una cadena con el patrón `Office(Department)`. Toma como entrada el nombre de la oficina, el nombre del departamento y dos constantes que aportan el paréntesis de apertura y de cierre respectivamente (*véase* *Funciones básicas* 197).
2. Una variable intermedia. El papel que desempeña es recopilar todos los datos de cada empleado en el mismo contexto de asignación. La variable consigue que la asignación busque el departamento y la

oficina de cada empleado, en el contexto de cada empleado. Es decir, la variable *recuerda* el nombre de la oficina y del departamento al que pertenece cada empleado. Sin la variable, el contexto sería incorrecto y la asignación produciría resultados no deseados (*véase* *Reglas y estrategias de asignación de datos* 407). Tenga en cuenta que la variable reproduce la estructura del archivo XML de destino (usa el mismo esquema XML). Esto permite conectar el resultado del componente de ordenación con el componente de destino por medio de una conexión de copia total. Consulte las secciones Usar variables 160 y Conexiones de copia total 55 para obtener más información.

3. Un componente de ordenación que ordena los datos. Observe que el criterio de ordenación del componente está conectado al elemento `Last` de la variable, lo cual permite ordenar todos los registros por apellido.

5.5 Filtros y condiciones

Cuando necesite filtrar datos o conseguir un valor de forma condicional, utilice uno de estos componentes:

- Filtro: nodos/filas ( )
- Condición If-Else ( )

Estos componentes se pueden añadir en la asignación desde el menú Insertar o desde la barra de herramientas Insertar componente. A continuación vamos a explicar las diferencias entre estos componentes, su comportamiento y sus requisitos.

Filtrar nodos o filas

Cuando necesite filtrar datos, incluidos nodos XML, utilice un componente **Filtro: nodos/filas**. Este componente permite recuperar un subconjunto de nodos de un conjunto de datos de mayor tamaño, usando una condición true o false. Esta es su estructura en el área de asignación:

En la estructura de filtro anterior, la condición que está conectada a bool determina si la entrada **node/row** que está conectado se pasa a la salida **on-true** o a la salida **on-false**. Es decir, si la condición se cumple (es true), entonces **node/row** se redirige a la salida **on-true**. Por el contrario, si la condición no se cumple (es false), entonces **node/row** se redirige a la salida **on-false**.

Cuando necesite que la asignación solamente consuma elementos que *cumplan* la condición de filtrado, puede dejar sin conectar la salida **on-false**. Si necesita procesar los elementos que no cumplan la condición de filtrado, entonces conecte la salida **on-false** al nodo de destino donde se deben redirigir dichos elementos.

Para ver un ejemplo con instrucciones paso a paso consulte el apartado Ejemplo: filtrar nodos 179.

Devolver un valor de forma condicional

Si necesita obtener un solo valor (en lugar de un nodo o de una fila) de forma condicional, lo mejor es usar la **condición If-Else**. Recuerde que estas condiciones no son adecuadas para el filtrado de nodos o filas. A diferencia de los componentes **Filtro: nodos/filas**, las **condiciones If-Else** devuelven un valor de tipo simple (como una cadena o un entero). Por tanto, estas condiciones solamente deben utilizarse cuando sea necesario procesar un valor simple de forma condicional. Por ejemplo, imagine que tiene una lista de las temperaturas medias de cada mes en este formato:

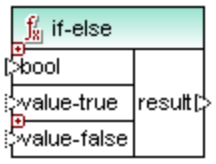
```
<Temperatures>
   <data temp="19.2" month="2010-06" />
   <data temp="22.3" month="2010-07" />
   <data temp="19.5" month="2010-08" />
   <data temp="14.2" month="2010-09" />
   <data temp="7.8" month="2010-10" />
   <data temp="6.9" month="2010-11" />
```



```
   <data temp="-1.0" month="2010-12" />
</Temperatures>
```

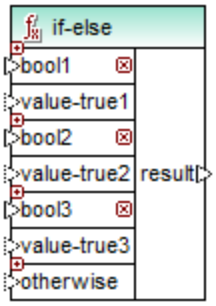
Una **condición If-Else** permite devolver por cada elemento de la lista el valor "alto" si la temperatura supera los 20 grados centígrados y el valor "bajo" si la temperatura es inferior a los 5 grados centígrados.

En la asignación la estructura de la **condición If-Else** sería:



Si la condición que está conectada a **bool** es `true`, entonces el valor que está conectado a **value-true** se devuelve como **result**. Si la condición es `false`, el valor que está conectado a **value-false** se devuelve como **result**. El tipo de datos de **result** se conoce previamente y depende del tipo de datos del valor que está conectado a **value-true** o **value-false**. Lo importante es que siempre sea de tipo simple (cadena, entero, etc.). Las **condiciones If-Else** no admiten el uso de valores de entrada de tipo complejo (como nodos o filas).

Además, las **condiciones If-Else** son extensibles. Esto significa que puede añadir varias condiciones al componente con solo hacer clic en el botón **Agregar** ⊡. Para eliminar una condición añadida previamente, haga clic en el botón **Eliminar** ⊠. Esta característica permite comprobar si se cumplen varias condiciones y devolver un valor distinto para cada condición.



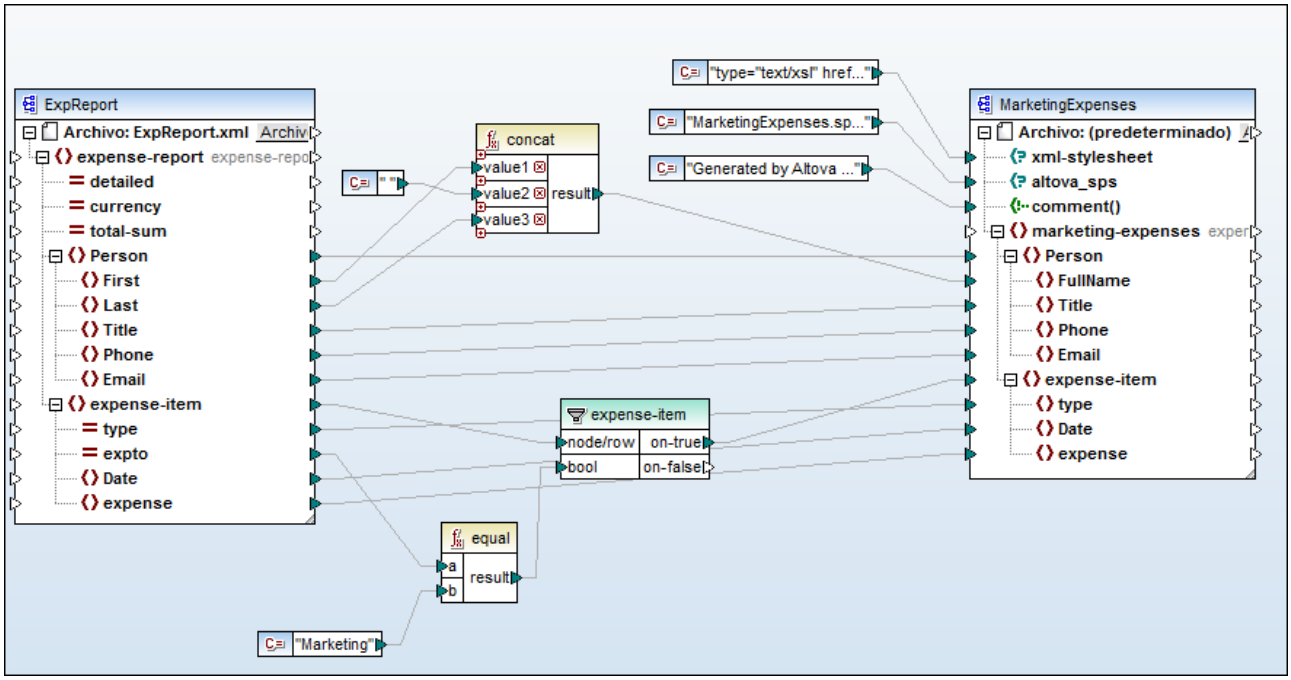
Las **condiciones If-Else** extendidas, se evalúan de arriba a abajo (se comprueba en primer lugar las condiciones situadas más arriba). Si desea devolver un valor aunque no se cumpla ninguna condición (es decir, aunque ninguna condición sea true), conecte el valor a **`otherwise`**.

Para ver un ejemplo con instrucciones paso a paso consulte el apartado Ejemplo: devolver un valor de forma condicional 181.

5.5.1 Ejemplo: filtrar nodos

Este ejemplo explica cómo filtrar nodos basándose en una condición true/false. Para conseguirlo se utiliza un componente **Filtro: nodos/filas** ( ).

La asignación de este ejemplo es **<Documentos>\Altova\MapForce2024\MapForceExamples\MarketingExpenses.mfd**.



Como puede verse en la imagen, la asignación lee datos de un XML de origen que contiene un informe de gastos ("ExpReport") y escribe datos en un XML de destino ("MarketingExpenses"). Entre el componente de origen y de destino existen varios componentes más. El más relevante en este ejemplo es el componente de filtrado **expense-item** ( ).

El objetivo de la asignación es filtrar los gastos que pertenecen al departamento de Marketing. Para conseguirlo se añadió un componente de filtrado a la asignación. (Para añadir un filtro haga clic en el menú Insertar y elija el comando **Filtro: nodos/filas**.)

Para identificar qué gastos pertenecen al departamento de Marketing, la asignación busca el valor del atributo "expto" del XML de origen. Este atributo tiene el valor "Marketing" si el gasto pertenece a dicho departamento. Por ejemplo, en el siguiente fragmento de código, puede verse que el primer gasto y el tercero pertenecen a Marketing, el segundo a Development (desarrollo) y el cuarto a Sales (ventas):

```
...
   <expense-item type="Meal" expto="Marketing">
      <Date>2003-01-01</Date>
      <expense>122.11</expense>
   </expense-item>
   <expense-item type="Lodging" expto="Development">
      <Date>2003-01-02</Date>
      <expense>122.12</expense>
   </expense-item>
   <expense-item type="Lodging" expto="Marketing">
      <Date>2003-01-02</Date>
      <expense>299.45</expense>
   </expense-item>
   <expense-item type="Entertainment" expto="Sales">
```



```
      <Date>2003-01-02</Date>
      <expense>13.22</expense>
   </expense-item>
...
```

XML de entrada antes de ejecutarse la asignación

En el área de asignación, la entrada **node/row** del filtro está conectado con el nodo **expense-item** del componente de origen. Esto permite al filtro obtener la lista de nodos que debe procesar.

Para agregar la condición en la que se debe basar el filtrado, se añadió la función **equal** de la biblioteca **core** de MapForce (*véase* *Funciones básicas* 197). La función **equal** compara el valor del atributo "type" con una constante cuyo valor es "Marketing". (Para agregar una constante haga clic en el menú **Insertar** y elija el comando **Constante**.)

Como nuestro objetivo es filtrar los elementos que cumplen esta condición, conectamos solamente la salida **on-true** del filtro con el componente de destino.

Cuando consultamos la vista previa del resultado de la asignación (en el panel *Resultados*), MapForce evalúa la condición conectada a la entrada **bool** del filtro en cada nodo expense-item. Si la condición se cumple (es true), el nodo expense-item se pasa al destino. De lo contrario, se pasa por alto. Como resultado se obtienen los gastos que cumplen los criterios:

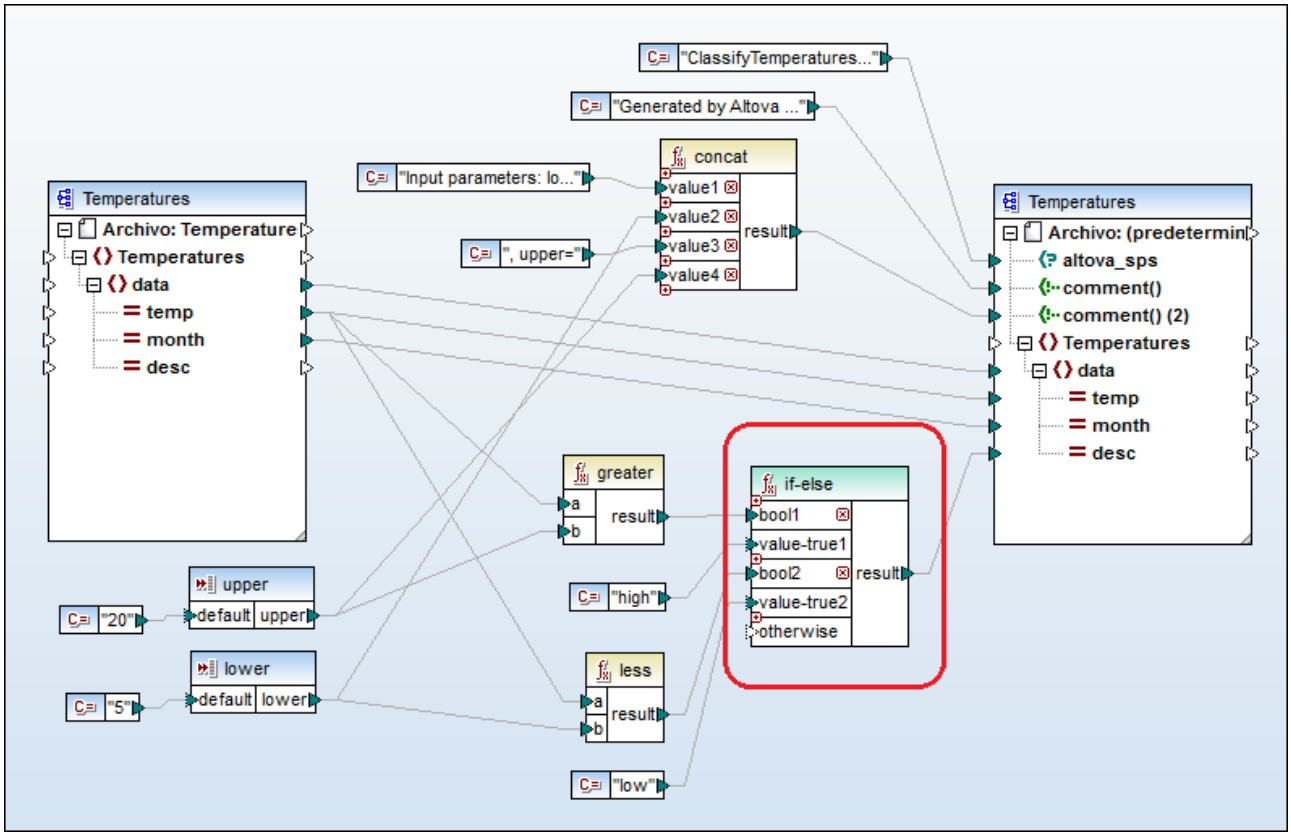
```
...
   <expense-item>
      <type>Meal</type>
      <Date>2003-01-01</Date>
      <expense>122.11</expense>
   </expense-item>
   <expense-item>
      <type>Lodging</type>
      <Date>2003-01-02</Date>
      <expense>299.45</expense>
   </expense-item>
...
```

XML de salida después de ejecutarse la asignación

5.5.2 Ejemplo: devolver un valor de forma condicional

Este ejemplo explica cómo obtener un valor simple de un componente con ayuda de una condición true/false. Para conseguirlo se utiliza una **condición If-Else** ( ). Las **condiciones If-Else** no se deben confundir con los componentes de filtrado. Las primeras son adecuadas para procesar valores simples de forma condicional (cadenas, enteros, etc.). Los componentes de filtrado, sin embargo, deben utilizarse para filtrar valores complejos como nodos (*véase* *Ejemplo: filtrar nodos* 179).

La asignación de este ejemplo es **<Documentos>\Altova\MapForce2024\MapForceExamples\ClassifyTemperatures.mfd**.



Esta asignación lee datos de un XML de origen que contiene datos sobre temperaturas ("Temperatures") y escribe datos en un XML de destino que cumple el mismo esquema. Entre el destino y el origen existen varios componentes más. De ellos el más relevante es la condición if-else (marcada en rojo).

El objetivo de la asignación es agregar una breve descripción a cada registro de temperatura en el documento de destino. Concretamente, si la temperatura supera los 20 grados centígrados, la descripción será "high" (alta). Si, por el contrario, la temperatura no supera los 5 grados centígrados, la descripción será "low" (baja). En el resto de los casos, no se incluirá ninguna descripción.

Para conseguir nuestro objetivo es necesario un procesamiento condicional y, por tanto, se añadió una condición if-else a la asignación. (Para añadir una condición if-else haga clic en el menú **Insertar** y después elija el comando **Condición If-Else**). En esta asignación la condición if-else se amplió (con ayuda del botón ⊡) para que incluyera dos condiciones: **bool1** y **bool2**.

Las condiciones propiamente dichas vienen dadas por las funciones **`greater`** y **`less`**, que se añadieron desde la biblioteca **core** de MapForce (*véase* *Trabajar con funciones* 197). Estas funciones evalúan los valores que aportan dos componentes de entrada llamados "upper" y "lower". (Para añadir un componente de entrada haga clic en el menú Insertar y elija el comando Insertar componente de entrada. Consulte el apartado Supplying Parameters to the Mapping 147 para obtener más información.)

Las funciones **`greater`** y **`less`** devuelven el valor true o false. El resultado de la función determina qué texto se escribe en la instancia de destino. Es decir, si el valor del atributo "temp" del XML de origen es superior a 20, entonces se pasa el valor de constante "high" a la condición if-else. Si el valor del atributo "temp" del XML de origen es inferior a 5, entonces se pasa el valor de constante "low" a la condición **if-else**. La entrada

otherwise de la condición se deja sin conectar. Por tanto, si no se cumple ninguna de las condiciones anteriores, no se pasará nada al conector de salida **result**.

Por último, el conector de salida **result** suministra este valor (uno por cada registro de temperatura) al atributo "desc" del componente de destino.

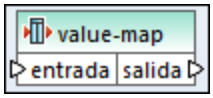
Para consultar una vista previa del resultado de la asignación haga clic en el panel *Resultados*. Observe que el XML de salida resultante incluye ahora el atributo "desc" cuando la temperatura es superior a 20 o inferior a 5.

```
...
   <data temp="-3.6" month="2006-01" desc="low"/>
   <data temp="-0.7" month="2006-02" desc="low"/>
   <data temp="7.5" month="2006-03"/>
   <data temp="12.4" month="2006-04"/>
   <data temp="16.2" month="2006-05"/>
   <data temp="19" month="2006-06"/>
   <data temp="22.7" month="2006-07" desc="high"/>
   <data temp="23.2" month="2006-08" desc="high"/>
...
```

XML de salida después de ejecutarse la asignación

5.6 Asignación de valores

Con los componentes value-map (*imagen siguiente*) puede reemplazar un valor por otro con la ayuda de una tabla de consulta predefinida. Este tipo de componente solo procesa un valor cada vez, por lo que en la asignación tiene una **entrada** y una **salida**.



Los componentes value-map son útiles si quiere asignar elementos individuales dentro de dos conjuntos para reemplazar elementos. Por ejemplo, puede expresar los días de la semana como números (1, 2, 3, 4, 5, 6 y 7) y asignarlos a los nombres de los días ("lunes", "martes", "miércoles", etc.). También puede asignar los nombres de los meses ("enero", "febrero", "marzo", etc.) a la representación numérica de los mismos (1, 2, 3, 4, etc.). Cuando ejecute la asignación los valores que coincidan serán reemplazados según su tabla personal de consulta. Los valores de estos dos conjuntos pueden ser de distinto tipo, pero cada conjunto debe almacenar valores del mismo tipo de datos.

Los componentes value-map sirven para realizar consultas simples en las que cada valor del primer conjunto corresponde a un único valor del segundo conjunto. Si no se encuentra algún valor en la tabla de consulta puede optar por reemplazarlo con el valor que quiera, con un valor vacío o pasarlo tal y como esté. Si necesita buscar o filtrar valores en base a un criterio más complejo, es mejor que use uno de los componentes de filtrado 178.

Es importante subrayar que cuando se genera código o se compila un archivo de ejecución de MapForce Server desde la asignación, la tabla de consulta está incrustada en el código o archivo generado. En consecuencia, solo recomendamos que defina una tabla de consulta directamente en la asignación si los datos de esa tabla no cambian a menudo y si no se trata de una gran cantidad de datos (menos de unas cien entradas). Si los datos de la tabla de consulta cambian a menudo, puede que resulte difícil mantener al día tanto la asignación como el código generado. En este caso sería más sencillo guardar la tabla de consulta en formato texto, XML, BD o Excel.

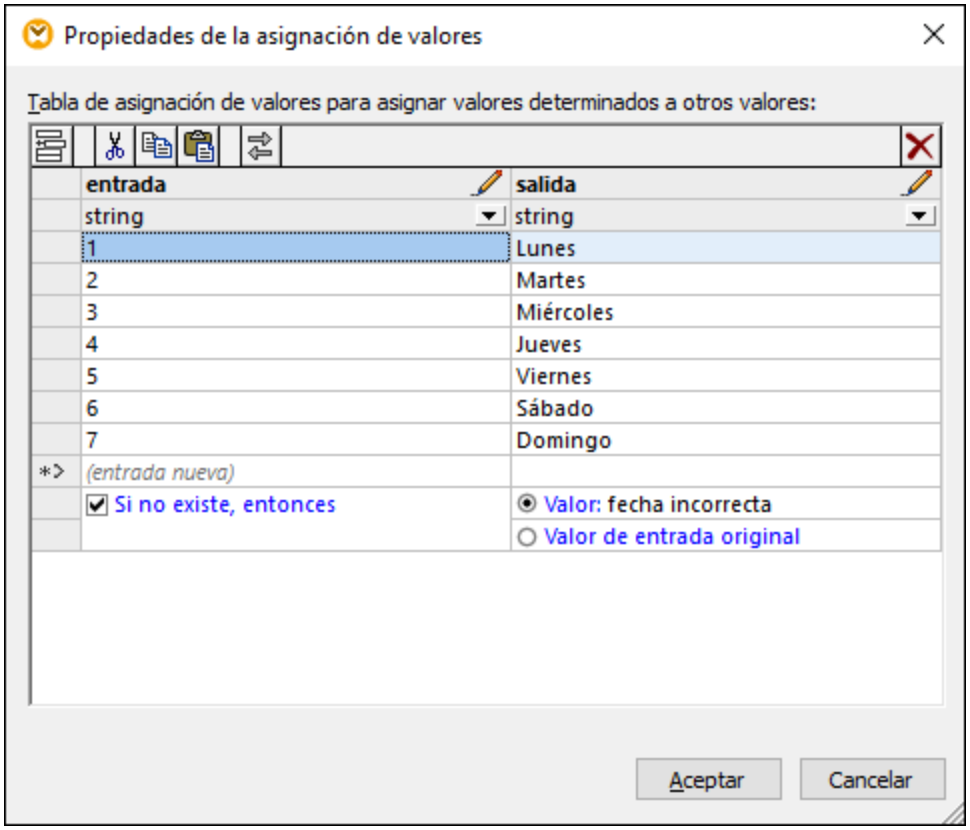
Si la tabla de consulta es enorme ralentizará la ejecución de la asignación. En este caso es mejor utilizar un componente de BD con SQL-Where. Los componentes de BD están disponibles tanto en la edición MapForce Professional como MapForce Enterprise. En estos casos son útiles las bases de datos SQLite, dada su portabilidad. Del lado servidor puede mejorar el rendimiento de las tablas de consulta ejecutando sus asignaciones con MapForce Server o MapForce Server Advanced Edition.

Crear una asignación de valores

Para añadir un componente de asignación de valores a la asignación, elija una de estas opciones:

- Haga clic en el botón de la barra de herramientas **Insertar asignación de valores**  .
- En el menú **Insertar** haga clic en **Asignación de valores**.
- Haga clic con el botón derecho en una conexión y seleccione **Insertar asignación de valores** en el menú contextual.

Así añade un componente de asignación de valores nuevo a la asignación. Ahora puede empezar añadir pares de elementos a la tabla de consulta. Para ello haga doble clic en la barra del título del componente o haga clic con el botón derecho y seleccione **Propiedades** en el menú contextual.



En el momento de ejecutar la asignación, MapForce comprueba cada valor que llega a la **entrada** del componente value-map. Si hay un valor que coincida *en la columna izquierda* de la tabla, este reemplaza al valor de entrada original con el valor *de la columna derecha*. De lo contrario puede configurarla para que devuelva:

- Un valor de reemplazo. En el ejemplo anterior el valor de reemplazo es el texto "fecha incorrecta". También puede definir el valor de reemplazo como un valor vacío. Para ello basta con que no introduzca ningún texto en ese campo.
- El valor de entrada original. Esto significa que si no hay ninguna coincidencia en la tabla de consulta el valor que se pasa a la asignación es el valor de entrada original sin modificar.

Si no configura una condición "Si no existe, entonces", entonces el componente value-map devuelve un **nodo vacío** si no encuentra una coincidencia. En este caso no se pasa ningún valor al componente de destino y el resultado contiene campos vacíos. Para evitar esto debe configurar la condición "Si no existe, entonces" o usar la función **`substitute-missing`**[307].

No es lo mismo definir un valor de reemplazo como vacío y no definir la condición "Si no existe, entonces". En el primer caso, el campo se genera en el resultado, pero contiene un valor vacío. En el segundo caso, el campo (o elemento XML) que contiene el valor ni siquiera se crea. Para más información consulte Ejemplo: reemplazar puestos de trabajo[192].

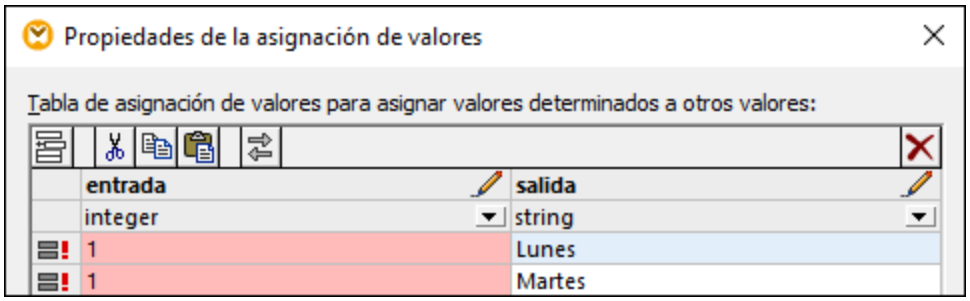
Rellenar una asignación de valores

En una tabla de consulta puede definir tantos pares de valores como necesite. Puede introducirlos de forma manual o copiar los datos de la tabla desde archivos de texto, CSV o Excel. En la mayoría de los casos también se pueden copiar y pegar tablas desde una página HTML usando un navegador web. Si copia datos desde archivos de texto los campos deben estar separados por el carácter tabulador. Por lo general MapForce también reconoce texto separado por coma o por punto y coma.

Al crear tablas de consulta, tenga en cuenta lo siguiente:

1. Todos los elementos de la columna izquierda deben ser únicos. De lo contrario no es posible determinar qué elemento en concreto quiere asignar.
2. Los elementos que pertenecen a la misma columna deben ser del mismo tipo de datos. Puede elegir el tipo de datos de la lista desplegable que hay al principio de cada columna en la tabla de consulta. Si necesita convertir tipos booleanos, introduzca el texto "true" o "false". Para ver un ejemplo consulte el apartado Ejemplo: reemplazar días de la semana 189.

Si MapForce encuentra datos no válidos en la tabla de consulta, entonces muestra un mensaje de error y resalta las filas no válidas en color rosa:



Para usar una fuente de datos externa para importar datos en el componente de asignación de valores, siga estos pasos:

1. Seleccione las celdas relevantes en el programa de origen (por ejemplo, Excel). Puede seleccionar una sola columna de datos o dos adyacentes.
2. Copie los datos en el portapapeles con el comando **Copiar** del programa externo.
3. En el componente de asignación de valores haga clic en la fila anterior a la fila en la que quiere pegar los datos.
4. Haga clic en el botón **Pegar tabla desde el portapapeles** del componente de asignación de valores. También pude pulsar **Ctrl+V** o **Mayús+Insertar**.

Nota: El botón **Pegar tabla desde el portapapeles** sólo se habilita si copia primero datos de alguna fuente de datos (es decir, si hay datos en el portapapeles).

Si el portapapeles contiene varias columnas, en la tabla de consulta solamente se insertan las dos primeras columnas y el resto de ellas se omite. Si pega datos desde una única columna encima de otros valores aparece un menú contextual que pregunta si los datos del portapapeles se deben insertar como filas nuevas o si se deben sobrescribir las que ya existen. Por lo tanto, si lo que quiere es sobrescribir los datos de la tabla de consulta en vez de insertar nuevas filas, debe asegurarse de que el portapapeles debe contener una sola columna y no varias.

Para insertar filas manualmente antes de una fila que ya existe, primero debe hacer clic en la fila en cuestión y después pulsar el botón **Insertar** .

Para mover una fila que ya existe a una posición distinta, arrástrela hasta la nueva posición (hacia arriba o hacia abajo) mientras mantiene pulsado el botón izquierdo del ratón.

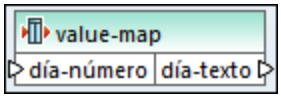
Si quiere copiar o cortar filas para después pegarlas en una posición distinta, primero seleccione la fila en cuestión y después haga clic en el botón **Copiar** (o **Cortar**). También puede copiar o cortar filas que no sean consecutivas. Para ello, mantenga pulsada la tecla **Ctrl** mientras va haciendo clic en las filas que quiere seleccionar. Tenga en cuenta que el texto cortado o copiado siempre contiene valores de ambas columnas, no puede cortar ni copiar valores de una columna solamente.

Para eliminar una fila haga clic en ella y después pulse el botón **Eliminar** .

Para intercambiar una columna por otra haga clic en el botón **Intercambiar columnas** .

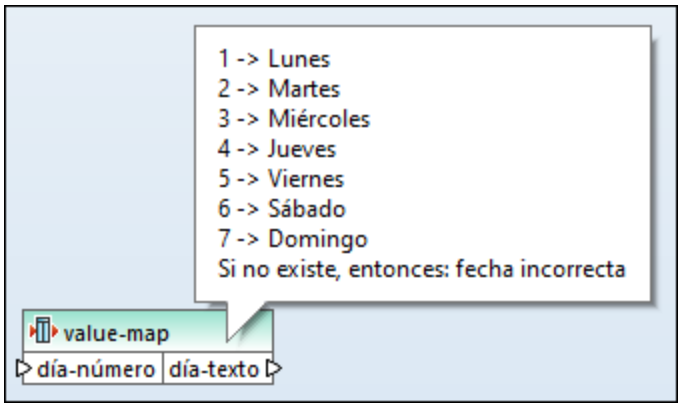
Cambiar el nombre de parámetros de asignación de valores

Por defecto, el parámetro de entrada de un componente de asignación de valores se llama "input" (entrada) y el parámetro de salida se llama "result" (resultado). Para que la asignación resulte más clara puede cambiar el nombre a cualquiera de estos parámetros haciendo clic en el botón **Editar** que hay junto al nombre en cuestión. Este es un ejemplo de un componente de asignación de valores en el que se han cambiado los nombres de los parámetros:



Vista previa de una asignación de valores

Una vez haya terminado de crear la asignación de valores puede acceder a una vista previa de su implementación directamente desde la asignación. Para ello sólo debe pasar el cursor por encima de la barra del título del componente:

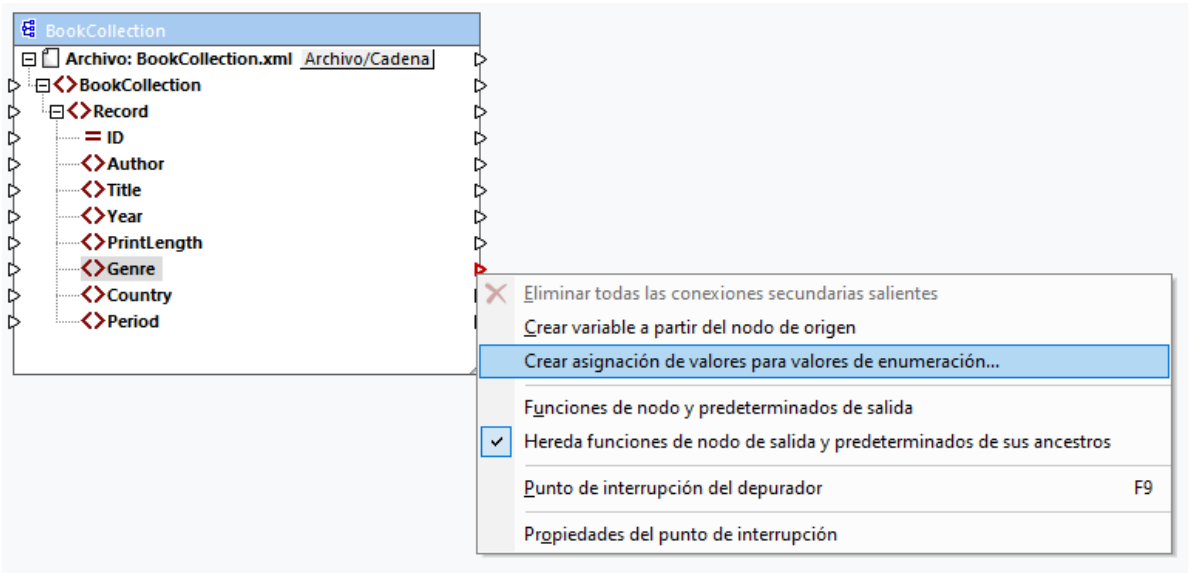


Crear una asignación de valores a partir de un valor de enumeración

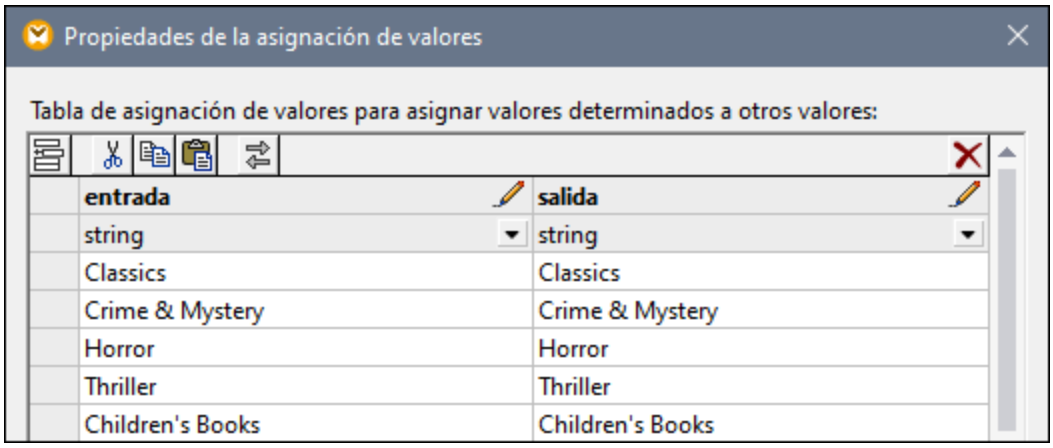
MapForce le permite crear una asignación de valores a partir de nodos con valores de enumeración. Actualmente, esta función es compatible con los componentes XML cuyos nodos tienen facetas de enumeración (*todas las ediciones*) y con los componentes EDI cuyos nodos tienen listas de códigos EDI (*edición Enterprise*). Puede crear este tipo de asignación de valores a partir del conector de entrada o de salida de un nodo, según lo que necesite.

Para crear una asignación de valores a partir de un valor de enumeración, siga estos pasos:

1. Haga clic con el botón derecho en el conector de entrada o salida del nodo (según sus objetivos) para los valores de enumeración desde los que desea crear una asignación de valores. En nuestro ejemplo (*imagen siguiente*) hemos seleccionado el conector de salida del nodo `Genre`.



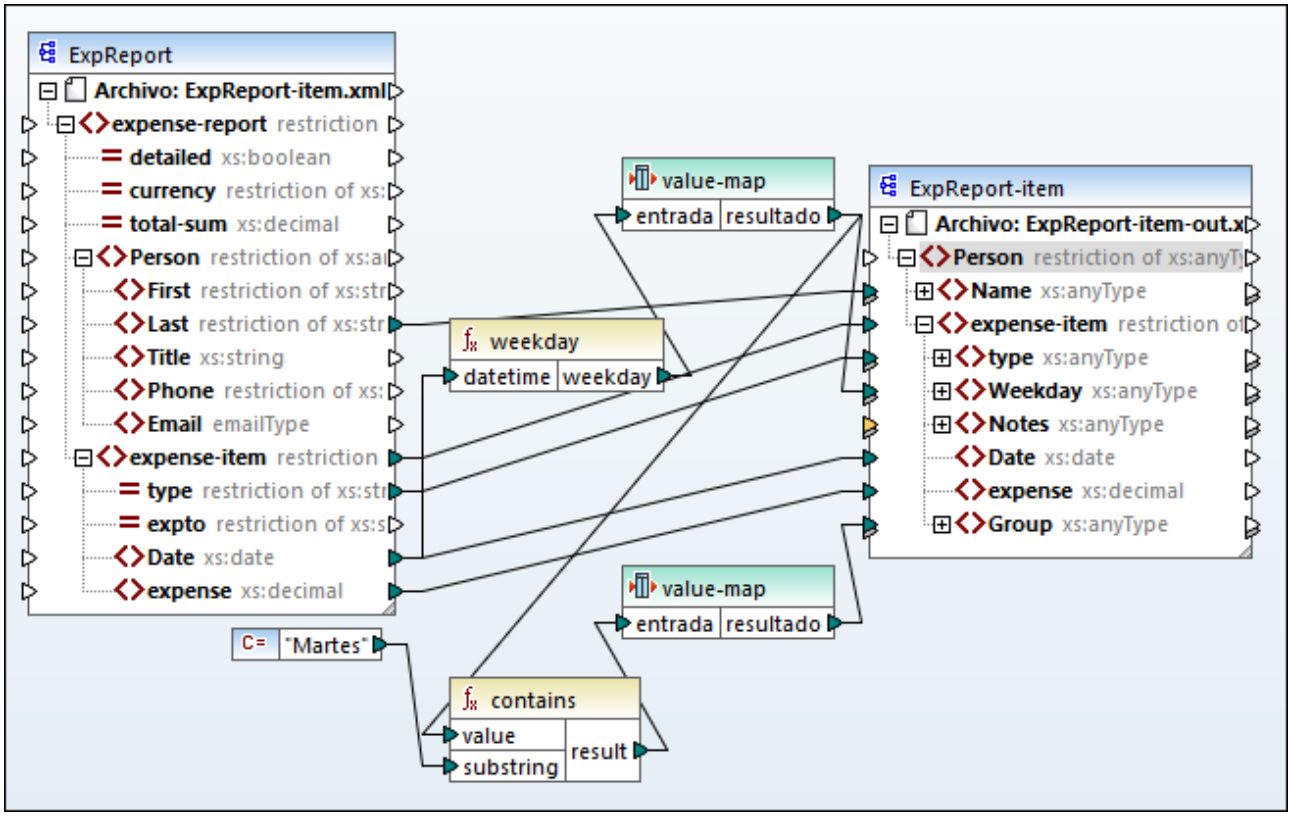
2. Seleccione **Crear asignación de valores para valores de enumeración** en el menú contextual.
3. Ahora aparecerá el cuadro de diálogo **Propiedades de la asignación de valores**. Tanto la parte de `entrada` como la de `salida` de la asignación de valores se prellenan con los mismos valores de enumeración. Ahora puede revisar y editar los valores según sea necesario. La siguiente imagen muestra la lista de valores `Genre` que tiene inicialmente el archivo XML de origen (`entrada`) y la lista de valores modificados que queremos asignar (`salida`).



4. Cuando termine de revisar los valores de enumeración, haga clic en **Aceptar**. Con esta acción se añadirá un componente value-map al área de asignación. El componente value-map se conectará automáticamente al nodo con cuyos valores de enumeración se creó la asignación de valores.
5. Conecte el otro parámetro de la asignación de valores con el nodo correspondiente y continúe diseñando su asignación como desee.

5.6.1 Ejemplo: reemplazar días de la semana

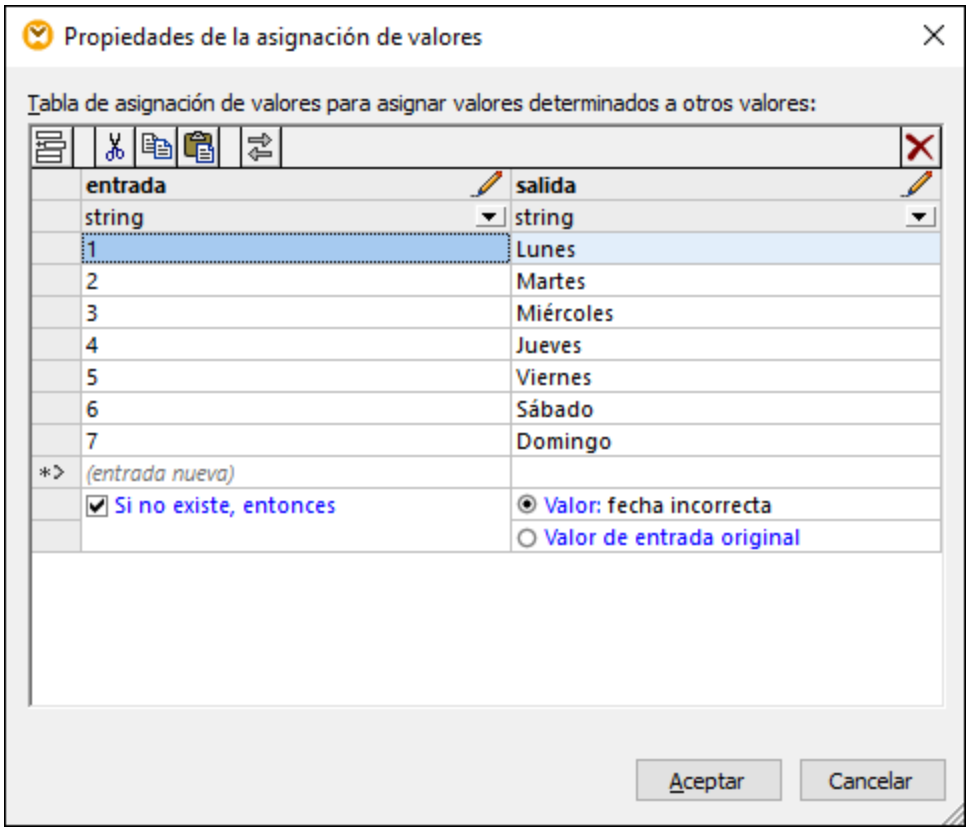
Este ejemplo consiste en una asignación de valores que reemplaza valores enteros con nombres de los días de la semana (1 = lunes, 2 = martes, etc.). El ejemplo viene acompañado de una asignación que puede encontrar en: **<Documentos>\Altova\MapForce2024\MapForceExamples\Tutorial\Expense-valmap.mfd**.



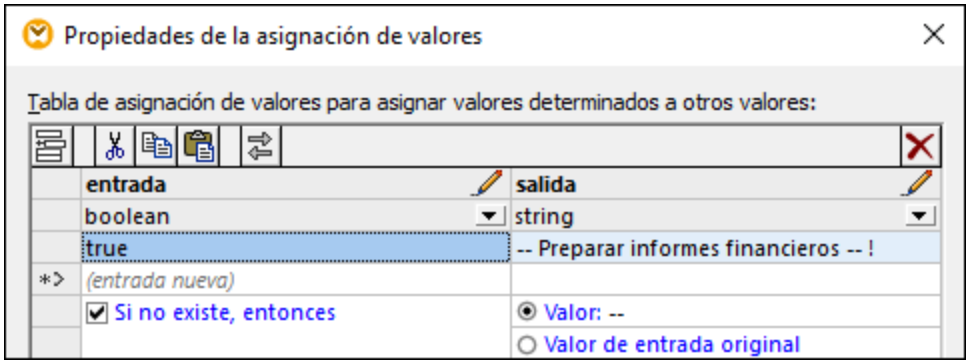
Expense-valmap.mfd

Esta asignación extrae el día de la semana del elemento **Date** en el archivo de origen, convierte el valor numérico en texto y lo escribe en el elemento **Weekday** del componente de destino. En concreto, ocurre lo siguiente:

- La función **weekday** extrae el número del día de la semana del elemento Date en el archivo de origen. El resultado de esta función son números enteros del 1 al 7.
- El primer componente de asignación de valores transforma los números enteros en días de la semana (1 = domingo, 2 = lunes). Si el componente se encuentra con un número entero no válido fuera del rango del 1 al 7, entonces devuelve el texto "fecha incorrecta".



- Si el día de la semana contiene "martes", entonces se escribe el texto "Preparar informes financieros" en el elemento **Notas** del componente de destino. Esto se consigue gracias a la función `contains`, que pasa un valor booleano **true** o **false** al segundo componente de asignación de valores. Ese segundo componente está configurado como se muestra en la imagen:



La asignación de valores de la imagen anterior se debe entender así:

- Siempre que se encuentre un valor booleano **true** se convierte en el texto "-- Preparar informes financieros -- ! ". En el resto de casos el texto es solamente "--".

Tenga en cuenta que el tipo de datos de la primera columna está definido como "boolean", lo que garantiza que el valor booleano **true** se reconozca como tal.

5.6.2 Ejemplo: reemplazar puestos de trabajo

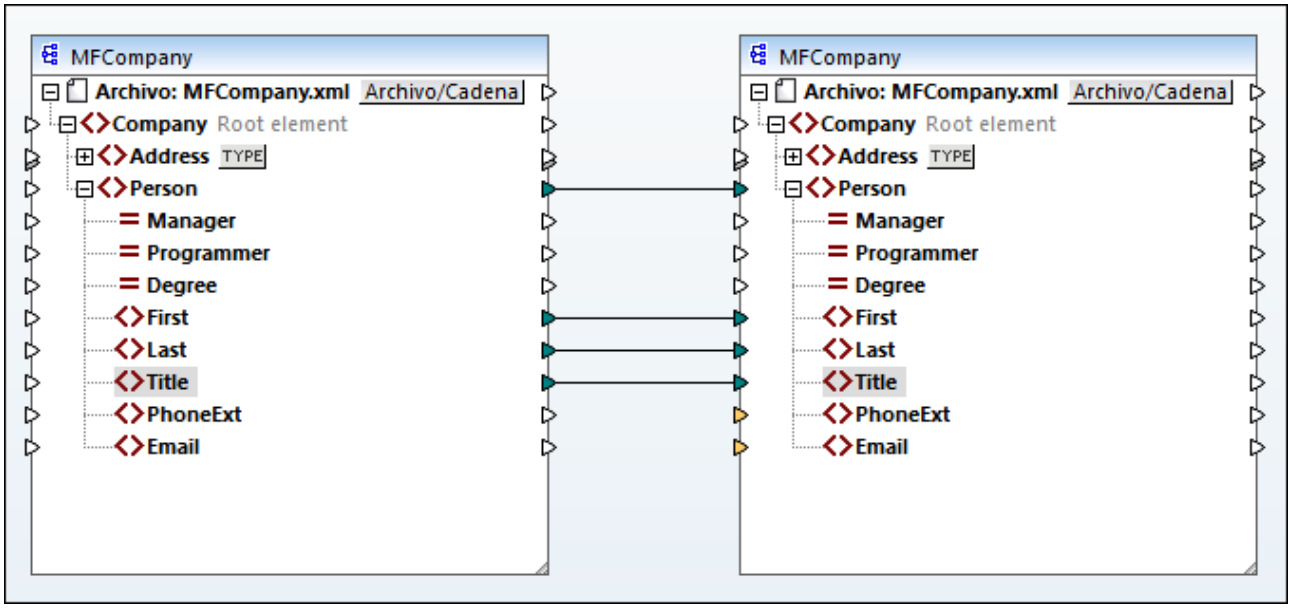
En este ejemplo aprenderá cómo reemplazar valores de elementos específicos en un archivo XML con ayuda de componentes de asignación de valores (es decir, usando una tabla de consulta predefinida).

Puede encontrar el archivo XML necesario para este ejemplo en **<Documentos>\Altova\MapForce2024\MapForceExamples\Tutorial\MFCompany.xml**. Además de otros datos, este archivo almacena información sobre los empleados de la empresa y sus puestos de trabajo, por ejemplo:

```
<Person>
   <First>Michelle</First>
   <Last>Butler</Last>
   <Title>Software Engineer</Title>
</Person>
<Person>
   <First>Lui</First>
   <Last>King</Last>
   <Title>Support Engineer</Title>
</Person>
<Person>
   <First>Steve</First>
   <Last>Meier</Last>
   <Title>Office Manager</Title>
</Person>
```

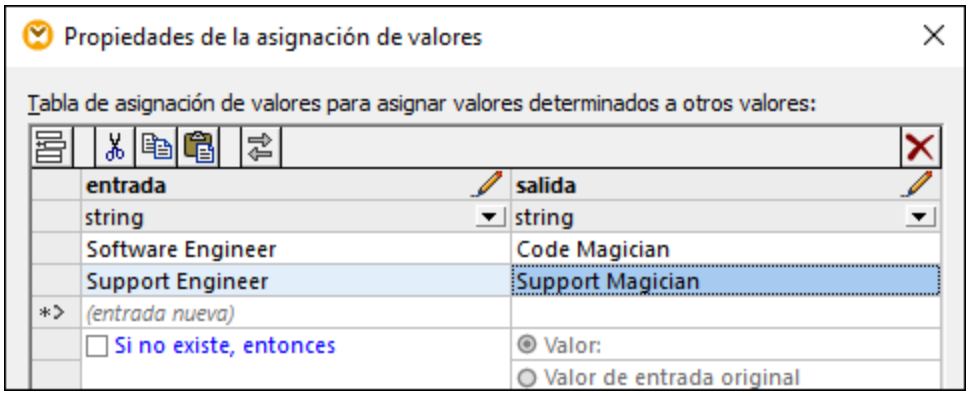
Imagine que necesita reemplazar algunos de esos puestos de trabajo en el archivo XML que hemos mencionado más arriba. En concreto, debe reemplazar "Software Engineer" (Ingeniero de software) por "Code Magician" (Mago del código). También el título "Support Engineer" (Ingeniero de soporte) se debe sustituir por "Support Magician" (Mago del soporte). El resto de puestos de trabajo permanece sin cambios.

Para conseguir este objetivo debe añadir el archivo XML al área de asignación: haga clic en el botón de la barra de herramientas **Insertar archivo o esquema XML**  o ejecutando el comando de menú **Insertar archivo o esquema XML**. A continuación, copie y pegue el componente XML en la asignación y cree las conexiones como se muestra en la imagen siguiente. Puede que primero necesite desactivar la opción Conectar automáticamente los secundarios equivalentes  para evitar que se creen automáticamente conexiones innecesarias.



La asignación que ha creado hasta ahora solamente copia los elementos **Person** en el archivo XML de destino sin hacer ningún cambio el los elementos **First**, **Last** o **Title**.

Para reemplazar los puestos de trabajo en cuestión debe añadir un componente de asignación de valores. Haga clic con el botón derecho en la conexión entre los dos elementos **Title** y seleccione **Insertar asignación de valores** en el menú contextual. Defina las propiedades de la asignación de valores como en la imagen:

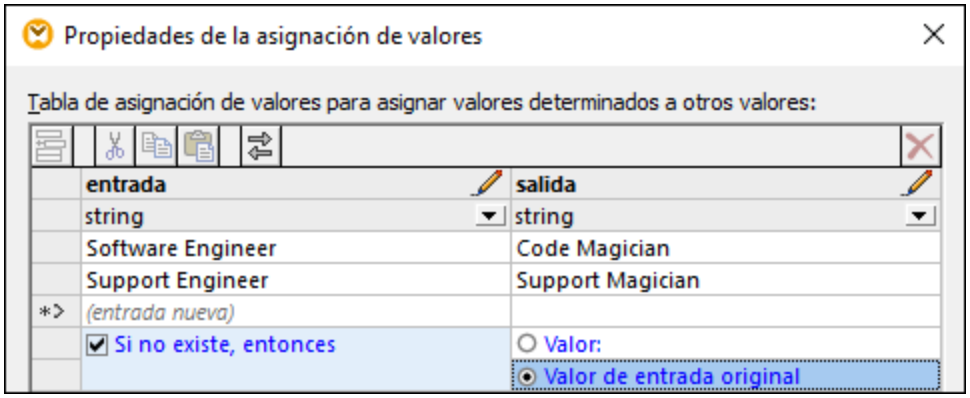


Según la configuración anterior, cada ocurrencia de "Software Engineer" se reemplaza con "Code Magician" y cada ocurrencia de "Support Engineer" se reemplaza con "Support Magician". Tenga en cuenta que no se ha definido la condición **Si no existe, entonces**, por lo que la asignación de valores devuelve un *nodo vacío* si el puesto de trabajo es distinto a "Software Engineer" o "Support Engineer". En consecuencia, si hace clic en la pestaña *Resultados* y accede a la vista previa de la asignación, en algunos de los elementos **Person** faltará el elemento **Title**, por ejemplo:

```
<Person>
   <First>Vernon</First>
```

```
   <Last>Callaby</Last>
</Person>
<Person>
   <First>Frank</First>
   <Last>Further</Last>
</Person>
<Person>
   <First>Michelle</First>
   <Last>Butler</Last>
   <Title>Code Magician</Title>
</Person>
```

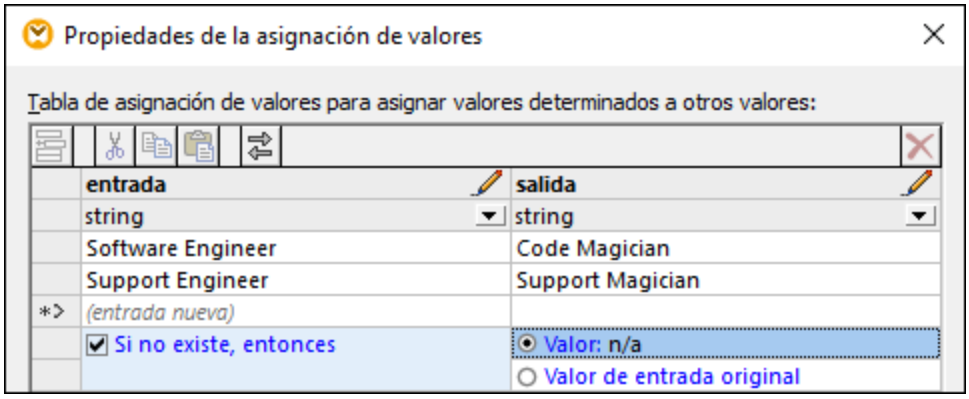
Como acabamos de explicar, si hay nodos vacíos en el resultado generado faltan algunas entradas; por tanto, en el fragmento XML anterior solamente se reemplazó el puesto de Michelle Butler, ya que este estaba en la tabla de consulta. La configuración que ha creado hasta ahora no se ajusta al objetivo inicial. Los ajustes correctos serían:



Con la configuración de la imagen anterior ocurre lo siguiente al ejecutar la asignación:

- Cada ocurrencia de "Software Engineer" se reemplaza con "Code Magician"
- Cada ocurrencia de "Support Engineer" se reemplaza con "Support Magician"
- Si el puesto original no se encuentra en la tabla de consulta, la asignación de valores lo devolverá sin cambios.

Sólo a efectos ilustrativos podemos cambiar también los puestos de trabajo que no sean "Software Engineer" o "Support Engineer" por un valor personalizado, por ejemplo "n/a". Para ello debe definir las propiedades de la asignación de valores como en la imagen siguiente:



Si ahora accede a una vista previa de la asignación, en el resultado aparecen todos los puestos de trabajo, pero aquellos que no coincidan con los dos que se han definido aparecen con el valor "n/a", por ejemplo:

```
<Person>
   <First>Vernon</First>
   <Last>Callaby</Last>
   <Title>N/A</Title>
</Person>
<Person>
   <First>Frank</First>
   <Last>Further</Last>
   <Title>N/A</Title>
</Person>
<Person>
   <First>Michelle</First>
   <Last>Butler</Last>
   <Title>Code Magician</Title>
</Person>
```

Aquí termina este ejemplo de asignación de valores. Puede usar este proceso lógico para conseguir el resultado deseado también en otras asignaciones.

6 Funciones

Las funciones suponen un potente mecanismo para transformar datos. En esta sección encontrará instrucciones para trabajar con funciones (tanto sin son funciones integradas de MapForce, como si son funciones definidas por el usuario o importadas de fuentes externas).

- **Funciones integradas de MapForce:** estas funciones vienen predefinidas en MapForce y se pueden usar en las asignaciones para tareas de procesamiento que conlleven cadenas de texto, números, fechas y otros tipos de datos. También puede usarlas para tareas de agrupamiento, suma, numeración automática y muchas tareas más. Para ver todas las funciones disponibles consulte la Referencia de la biblioteca de funciones 235.
- **Funciones definidas por el usuario:** estas son funciones de MapForce que puede crear usted mismo usando como base los tipos de componente nativos y las funciones integradas que ya están disponibles en MapForce (véase Funciones definidas por el usuario 205).
- **Funciones personales:** estas son funciones que puede importar de fuentes externas, como bibliotecas XSLT y adaptarlas a MapForce. Tenga en cuenta que para poder reutilizarlas en MapForce, los datos que devuelvan las funciones personalizadas deben ser de tipo simple (como una cadena o un número entero) y los parámetros que tomen también deben ser de tipo simple. Para más información consulte Importar funciones XSLT personales 223

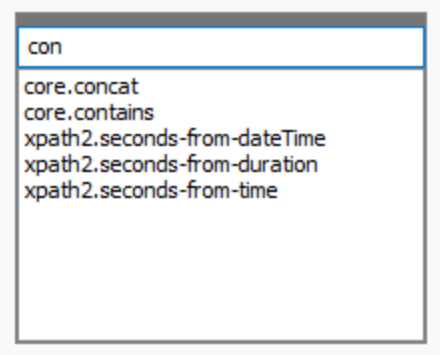
6.1 Fundamentos de las funciones

En las subsecciones siguientes encontrará un resumen de las acciones básicas relacionadas con funciones. Las funciones disponibles en la ventana Bibliotecas dependen del lenguaje de transformación seleccionado en ese momento. Para saber más consulte Seleccionar el lenguaje de transformación 17.

Agregar una función

MapForce cuenta con un amplio número de funciones integradas que puede agregar a sus asignaciones. Para más información sobre las funciones integradas disponibles consulte Referencia de la biblioteca de funciones 235. Hay varios métodos para agregar una función a una asignación:

- Pulse y mantenga pulsada la función que necesite en la ventana Bibliotecas y arrástrela hasta el área de asignación. Para filtrar las funciones por nombre empiece a teclear el nombre de la función en el cuadro de texto que hay en la parte inferior de la ventana.
- Haga doble clic en un área vacía de la asignación y empiece a teclear el nombre de la función (*ver imagen siguiente*). Para ver una ayuda rápida con información sobre la función vaya a la lista de funciones y haga clic en la función relevante. Para agregar una función a una asignación haga doble clic en la función relevante en el cuadro combinado.



Agregar funciones definidas por el usuario

También puede agregar funciones definidas por el usuario a sus asignaciones, bien como hemos explicado más arriba si (i) esa función definida por el usuario ya se ha creado en la misma asignación o (ii) si ha importado una asignación que contiene una función definida por el usuario como biblioteca local o global.

Agregar una constante

Las constantes permiten proporcionar a una asignación texto y cifras personales. Para agregar una constante a una asignación tiene varias opciones:

- Haga clic con el botón derecho en un área vacía de la asignación y seleccione **Insertar constante** en el menú contextual. Introduzca el valor y seleccione uno de estos tipos de datos: *cadena*, *número*, *demás opciones*.
- Añada una constante con el comando de menú **Insertar | Constante**. Introduzca el valor y seleccione uno de estos tipos de datos: *cadena*, *número*, *demás opciones*.
- Haga clic en el comando de la barra de herramientas **Constante**. Introduzca el valor y seleccione uno de estos tipos de datos: *cadena*, *número*, *demás opciones*.
- Haga doble clic en cualquier parte vacía del área de la asignación. Teclee comillas dobles seguidas del valor de la constante. No es necesario que cierre las comillas. Para añadir una constante numérica, simplemente teclee el número.

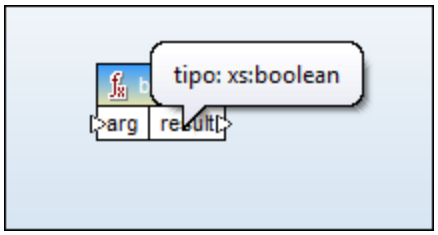
Buscar una función

Para buscar una función en la ventana **Bibliotecas** empiece a teclear el nombre de la función en el campo de texto que hay en la parte inferior de la ventana. Por defecto, MapForce busca el nombre de la función y el texto descriptivo. Si desea excluir la descripción de las funciones de la búsqueda, haga clic en el icono en forma de flecha y deshabilite la opción *Incluir descripción en la búsqueda*. Para cancelar la búsqueda pulse la tecla **Esc** o haga clic en el icono ✖.

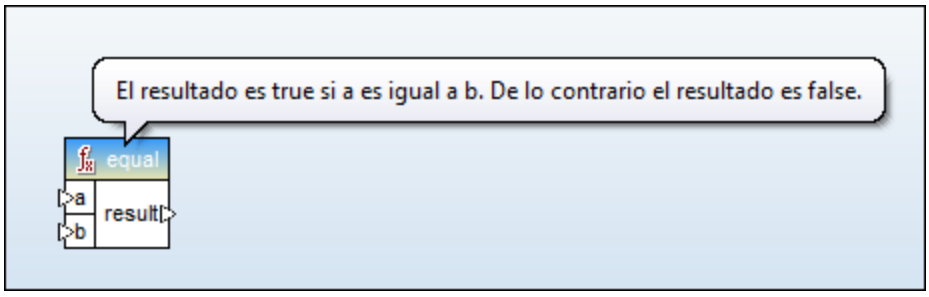
Para buscar todas las instancias de una función dentro de la asignación activa basta con hacer clic con el botón derecho en la función en la ventana **Bibliotecas** y elegir el comando **Buscar todas las llamadas** en el menú contextual. El resultado de la búsqueda aparece en la ventana **Mensajes**.

Ver el tipo y la descripción de una función

Para ver el tipo de datos del argumento de una función, pase el ratón por encima de la parte del argumento de una función (*imagen siguiente*). Asegúrese de que el botón (**Ver información rápida**) está habilitado.

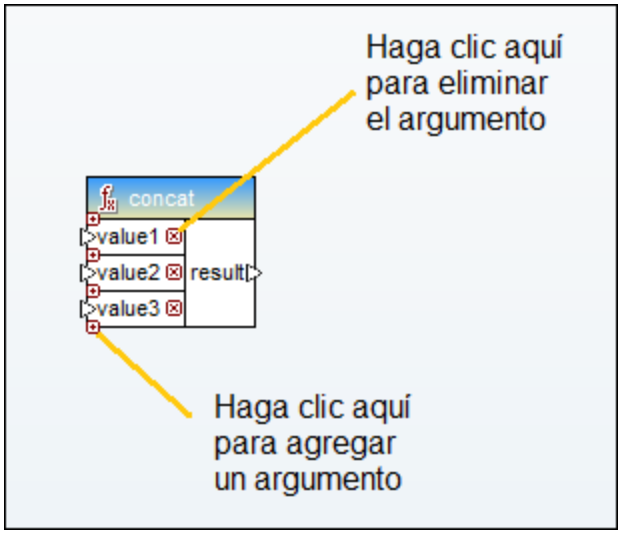


Para ver la descripción de una función mueva el ratón sobre el encabezado de la función (*imagen siguiente*). Asegúrese de que el botón (**Ver información rápida**) está habilitado.



Agregar o eliminar argumentos en una función

En algunas de las funciones integradas de MapForce es posible agregar tantos parámetros como necesite para la asignación. Un ejemplo de ello es la función **concat** 310. Para añadir o eliminar argumentos de una función (para las funciones que lo admiten) haga clic en **Agregar parámetro** (⊡) o **Eliminar parámetro** (⊠) junto al parámetro que quiere añadir o eliminar (*véase más abajo*). Si mueve una conexión al símbolo ⊡ se añade otro parámetro que se conecta a este.


Haga clic aquí
para eliminar
el argumento
concat
value1
value2
value3
result
Haga clic aquí
para agregar
un argumento

6.2 Gestionar bibliotecas de funciones

En MapForce puede importar y usar estos tipos de bibliotecas en una asignación:

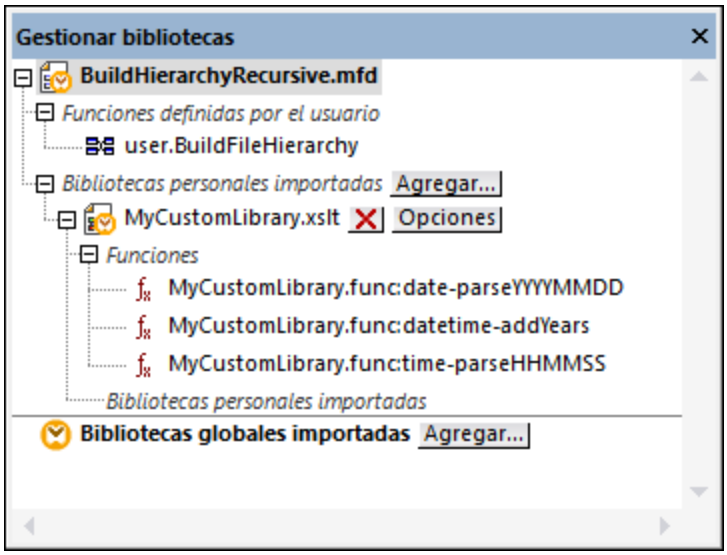
- Cualquier archivo de diseño de asignación (*.mfd) que contenga funciones definidas por el usuario. En concreto, asignaciones que contengan funciones definidas por el usuario creadas con MapForce, usando las funciones y los componentes integrados de MapForce como bloques de construcción. Para más información consulte Crear funciones definidas por el usuario 206.
- Archivos XSLT que contengan funciones. En concreto, funciones XSLT escritas fuera de MapForce pero que se pueden importar en MapForce como se describe en Importar funciones XSLT personales 223.

Ventana Gestionar bibliotecas

La ventana Gestionar bibliotecas permite ver y gestionar todas las bibliotecas que usa un archivo de asignación (tanto funciones definidas por el usuario como bibliotecas personales) al mismo tiempo.

Por defecto, la ventana Gestionar bibliotecas no está visible. Para que aparezca tiene dos opciones:

- En el menú **Vista** haga clic en **Gestionar bibliotecas**.
- En la parte inferior de la ventana Bibliotecas haga clic en **Agregar o quitar bibliotecas**.



Puede elegir si quiere ver solamente las funciones definidas por el usuario del archivo de asignación activo o si quiere ver las de todos los archivos de asignación abiertos. Para ver las funciones y bibliotecas importadas para todas las asignaciones abiertas actualmente haga clic dentro de la ventana y seleccione **Mostrar documentos abiertos** en el menú contextual.

Si en vez de el nombre del archivo de asignación abierto prefiere ver su ruta de acceso haga clic dentro de la ventana y seleccione **Mostrar todas las rutas de acceso** en el menú contextual.

Los datos que aparecen en la ventana Gestionar bibliotecas se organizan como una estructura en árbol:

- Los archivos de asignación abiertos se muestran como entradas de nivel superior. Cada entrada tiene dos ramas: **Funciones definidas por el usuario** y **Bibliotecas personales importadas**.

  - La rama **Funciones definidas por el usuario** muestra las funciones definidas por el usuario que contenga el archivo de asignación.
  - La rama **Bibliotecas personales importadas** muestra las bibliotecas importadas *localmente* en el archivo de asignación actual. El término "bibliotecas" se refiere a otros archivos de asignación (archivos .mfd que contienen funciones definidas por el usuario), a bibliotecas personales externas escritas en XSLT 1.0, XSLT 2.0, XQuery 1.0*, Java*, C#* o a los archivos .mff mencionados con anterioridad. Tenga en cuenta que la estructura **Bibliotecas personales importadas** podría tener varios niveles de profundidad, ya que un archivo de asignación puede importar otros archivos de asignación como bibliotecas.
- La entrada **Bibliotecas globales importadas** incluye todas las bibliotecas que haya importado *globalmente*, a nivel de la aplicación. La estructura de los archivos .mfd también puede tener varios niveles de profundidad por las mismas razones que se explican en el punto anterior.

** Estos lenguajes sólo son compatibles con las ediciones MapForce Professional y Enterprise.*

Nota: las bibliotecas XSLT, XQuery, C# y Java pueden tener dependencias propias. Estas dependencias no aparecen en la ventana Bibliotecas.

Comandos del menú contextual

Puede aplicar distintas acciones a los objetos de la ventana Gestionar bibliotecas. Para ello debe hacer clic con el botón derecho en el objeto en cuestión y seleccionar una de estas opciones del menú contextual:

Comando	Descripción	Se aplica a
Abrir	Abre la asignación.	Asignaciones
Agregar	Abre una caja de diálogo en la que puede navegar hasta una biblioteca personal de funciones.	Bibliotecas propias importadas
Buscar función en la ventana Bibliotecas	Pasa a estar activa la ventana Bibliotecas, donde se selecciona la función que se está buscando.	Funciones
Cortar, Copiar, Eliminar	Estos comandos estándar de Windows se pueden usar solamente con las funciones definidas por el usuario. No puede copiar o pegar funciones de archivos XSLT externos o de otros tipos de bibliotecas.	Funciones definidas por el usuario
Pegar	Permite pegar en la biblioteca actual una función definida por el usuario que haya copiado previamente en el portapapeles.	Bibliotecas (Funciones definidas por el usuario)
Opciones	Abre un cuadro de diálogo en el que puede definir o cambiar las opciones de la biblioteca actual.	Bibliotecas

Comando	Descripción	Se aplica a
Mostrar todos los documentos abiertos	Si se activa esta opción, la ventana Gestionar bibliotecas muestra todas las asignaciones abiertas, lo que es útil si necesita copiar y pegar funciones de unas asignaciones a otras. De lo contrario, en esta ventana sólo aparece la asignación que esté activa.	Siempre
Mostrar todas las rutas de acceso	Si se activa esta opción, la ventana Gestionar bibliotecas muestra la ruta completa de los objetos que contiene. De lo contrario aparecen los nombres, pero no las rutas.	Siempre

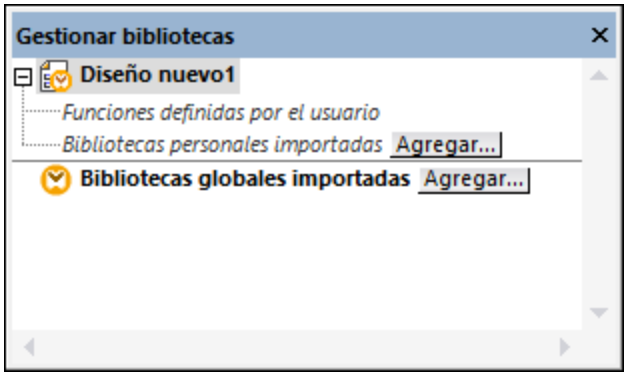
6.2.1 Bibliotecas locales y globales

Puede importar bibliotecas de forma local o global. Globalmente significa que importa la biblioteca a nivel de la aplicación. Si una biblioteca se importa de forma global, puede usar sus funciones para cualquier asignación.

Las importaciones a nivel local se hacen a nivel del archivo de asignación. Por ejemplo, imagine que está trabajando en la asignación **A.mfd** y decide importar todas las funciones definidas por el usuario de la asignación **B.mfd**. En este caso la asignación **B.mfd** se considera como una biblioteca importada localmente en **A.mfd**, por lo que puede usar funciones de **B.mfd** en **A.mfd**. Otro ejemplo de importación a nivel local sería importar funciones de un archivo XSLT en **A.mfd**.

Puede ver y gestionar todas las bibliotecas importadas desde la ventana Bibliotecas. Para importar una biblioteca:

1. Haga clic en el botón **Agregar o quitar bibliotecas**, en la parte inferior de la ventana Bibliotecas 21. Se abre la ventana **Gestionar bibliotecas**.



2. Para importar funciones como biblioteca local (dentro del archivo de asignación actual solamente), haga clic en **Agregar** bajo el nombre de la asignación activa. Para importar funciones como biblioteca global (a nivel de programa), haga clic en **Agregar**, junto a **Bibliotecas globales importadas**. Cuando se importa una biblioteca de forma local, puede hacer que la ruta de acceso al archivo de la biblioteca sea relativa al archivo de asignación. Con las bibliotecas globales la ruta siempre es absoluta.

Nombres de función conflictivos

Es posible que se encuentre en la situación de que un nombre de función esté definido en más de un nivel:

- en la asignación principal
- en una biblioteca importada localmente
- en una biblioteca importada globalmente

En estos casos, MapForce intentará llamar a la función en ese orden para evitar ambigüedades. Es decir, la función definida en la asignación tiene preferencia frente al mismo nombre de función definido en una biblioteca importada localmente. A su vez, la función importada localmente tiene preferencia frente a la función importada a nivel global (si ambas tienen el mismo nombre).

Si existen varias funciones con el mismo nombre, sólo se llama a la función "ganadora" según las reglas que acabamos de explicar. Los demás nombres ambiguos se bloquean y aparecen en gris en la ventana Bibliotecas, es decir, no se pueden usar en la asignación.

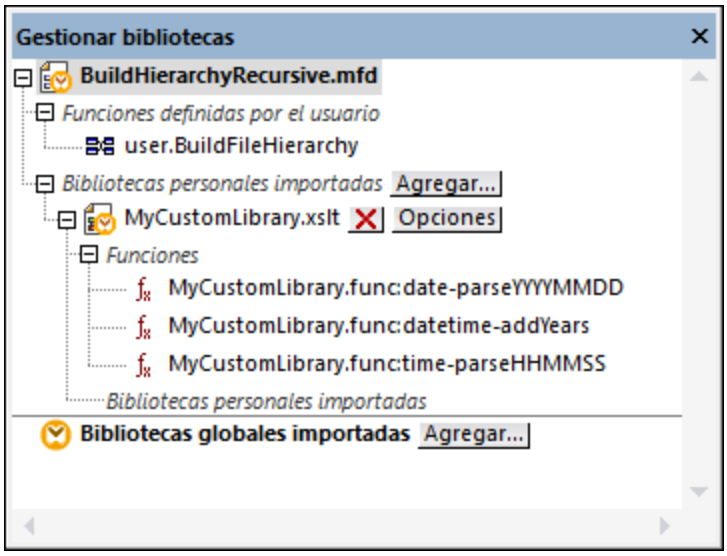
6.2.2 Rutas relativas de acceso a bibliotecas

Puede configurar la ruta de cualquier archivo de biblioteca que importe para que sea relativa al archivo de asignación (.mfd), siempre y cuando la biblioteca se haya importado de forma local (no global), como se describe en Bibliotecas locales y globales 202.

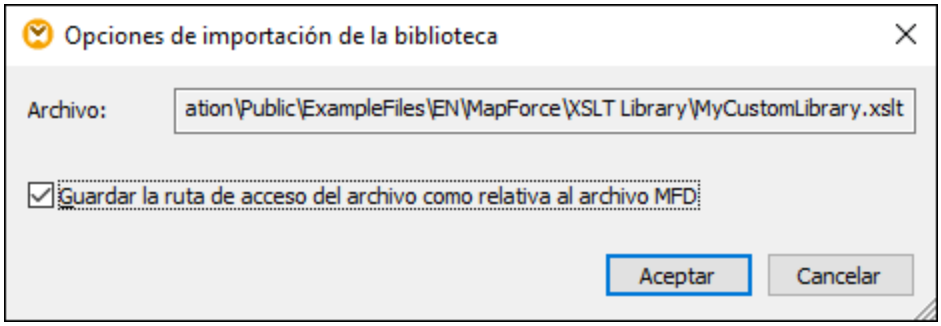
> Sólo se pueden configurar como relativas las rutas de acceso a bibliotecas que se hayan importado de forma *local* en una asignación. Si se importa una asignación *globalmente*, es decir, a nivel del programa, su ruta será siempre absoluta.

Para convertir la ruta de una biblioteca en relativa al archivo de asignación:

1. Haga clic en **Agregar o quitar bibliotecas**, en la parte inferior de la ventana Bibliotecas. Se abre la ventana Gestionar bibliotecas.



2. Haga clic en **Opciones**, junto a la biblioteca en cuestión. (También puede hacer clic con el botón derecho en la biblioteca y seleccionar **Opciones** en el menú contextual.)



3. Marque la casilla *Guardar la ruta de acceso del archivo como relativa al archivo MFD*.

Nota: si esta casilla aparece en gris, asegúrese de que importó la biblioteca de forna local y no global.

Cuando se marca esta casilla, MapForce llevará un registro y actualizará la ruta de cualquier archivo de biblioteca al que se haga referencia cuando guarde la asignación en un directorio nuevo usando el comando de menú **Guardar como**. Si los archivos de biblioteca están en el mismo directorio que la asignación, la referencia de la ruta no se romperá si mueve todo el directorio a una ubicación nueva del disco (*consulte también* *Usar rutas de acceso relativas en un componente* 42).

Tenga en cuenta que la casilla *Guardar la ruta de acceso del archivo como relativa al archivo MFD* indica que las rutas son relativas a la asignación, pero no afecta a las rutas en el código generado. Para saber más sobre las referencias a bibliotecas en el código generado consulte Rutas de acceso según el entorno de ejecución 44.

6.3 Funciones definidas por el usuario

Las funciones definidas por el usuario son funciones personales que se definen una vez y se pueden reutilizar varias veces dentro de la misma asignación o en varias asignaciones. Las funciones definidas por el usuario son como asignaciones de datos a pequeña escala: Suelen consistir en uno o más parámetros de entrada, algunos componentes intermediarios para procesar datos y una salida que devuelve datos al emisor de la llamada. El emisor de las llamadas es la asignación principal u otra función definida por el usuario.

Ventajas de las funciones definidas por el usuario

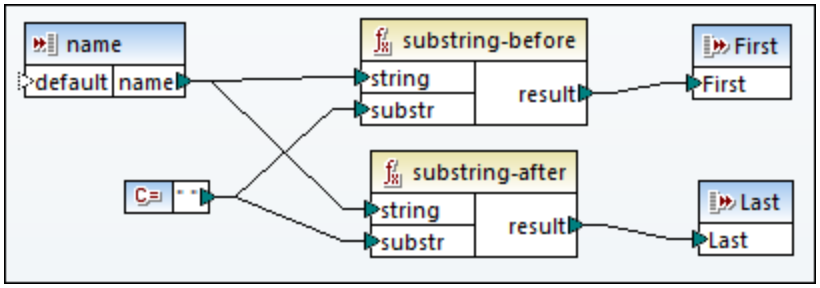
Las funciones definidas por el usuario tienen las ventajas siguientes:

- Se pueden reutilizar dentro de la misma asignación o en asignaciones distintas.
- Pueden hacer una asignación más legible. Por ejemplo, puede dividir partes de la asignación en componentes más pequeños y abstraer los detalles de la implementación. Puede observar este principio en el diagrama siguiente:

- Las funciones definidas por el usuario son funciones flexibles que permiten procesar cadenas, números, fechas y otros datos de forma personalizada y que amplía las funciones integradas de MapForce. Por ejemplo, puede que quiera concatenar o dividir texto de una forma concreta, hacer cálculos, modificar fechas y horas, etc.
- Otro uso común de las funciones definidas por el usuario es para buscar un campo en un archivo XML, una BD u otro formato de datos compatible con su edición de MapForce. Para más detalles consulte Implementación de la búsqueda 219.
- Puede llamar a las funciones definidas por el usuario de forma recursiva (por ejemplo, llamándose a sí misma). Para ello debe definir esa función definida por el usuario como función estándar (no inline) 208. Las funciones definidas por el usuario recursivas 216 permiten, entre otras tareas complejas de las asignaciones, iterar sobre estructuras de datos con una profundidad de *N* secundarios donde *N* no se conoce de antemano (véase el ejemplo Ejemplo: búsqueda recursiva).

Ejemplo

A continuación puede ver un ejemplo de una función definida por el usuario que divide una cadena en dos. Esta función definida por el usuario es parte de una asignación mayor llamada **`MapForceExamples\ContactsFromPO.mfd`**. La función definida por el usuario toma el nombre como parámetro (p.e., `Helen Smith`), aplica las funciones integradas **`substring-before`** y **`substring-after`** y devuelve dos valores: `Helen` y `Smith`.



Apartados de esta sección

En esta sección explicamos cómo trabajar con funciones definidas por el usuario; la sección sección se divide en estas subsecciones:

6.3.1 Funciones básicas definidas por el usuario

En esta sección explicamos cómo crear, importar, editar, copiar, pegar y eliminar funciones definidas por el usuario.

Crear una función definida por el usuario

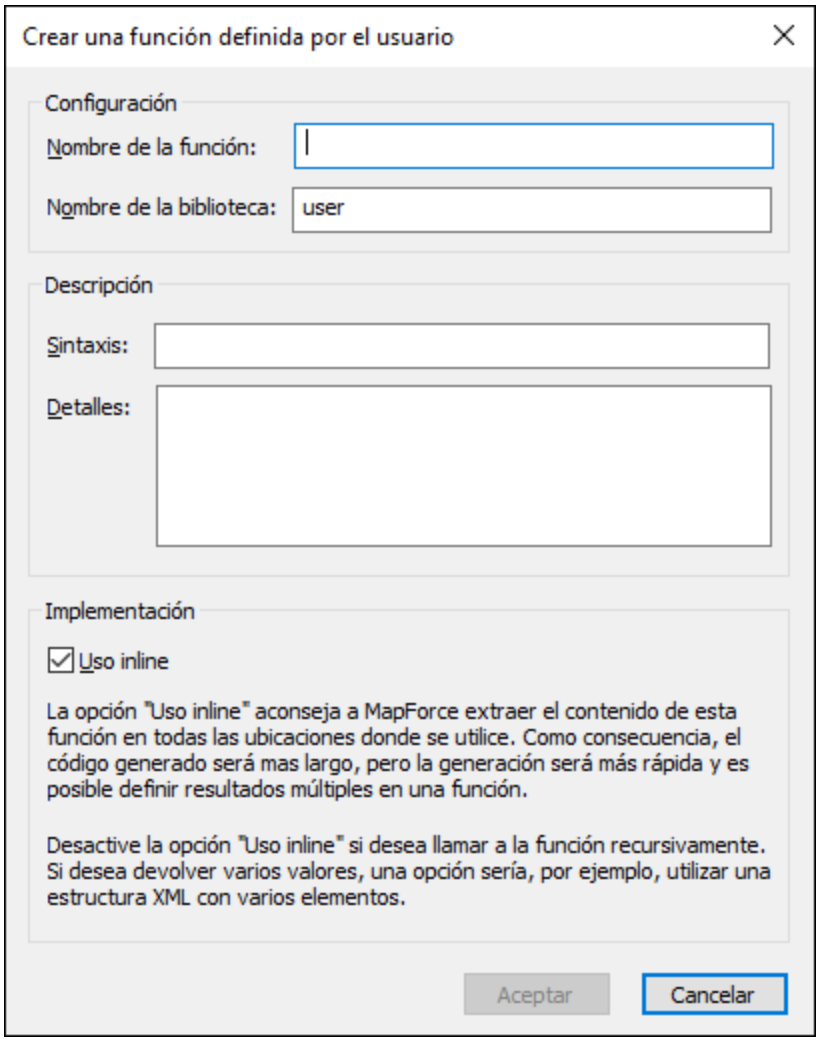
Aquí explicamos cómo crear una función definida por el usuario desde cero y a partir de componentes que ya existen. El requisito mínimo es un componente de salida al que haya datos conectados. En cuanto a los parámetros de entrada, una función puede no tener ninguno o tener uno o más. Los parámetros de entrada o de salida pueden ser de tipo simple (como una cadena de texto o un número entero) o de tipo complejo (una estructura). Para más información sobre parámetros simples y complejos, consulte el apartado Parámetros en funciones definidas por el usuario 211.

Crear una función definida por el usuario desde cero

Para crear una función definida por el usuario desde cero siga estos pasos:

1. Seleccione **Función | Crear una función definida por el usuario**. También puede hacer clic en el botón de la barra de herramientas.

2. Introduzca la información requerida en los campos correspondientes del cuadro de diálogo **Crear una función definida por el usuario** (*tabla siguiente*).



A continuación mostramos las opción disponibles.

- *Nombre de la función:* Campo obligatorio. Introduzca un nombre para la función definida por el usuario que quiere crear. Caracteres válidos: caracteres alfanuméricos (a-z, A-Z, 0-9), barra baja (_), guion (-) y dos puntos (:).
- *Nombre de la biblioteca:* Campo obligatorio. Introduzca el nombre de la biblioteca a la que debe pertenecer la función (en la ventana Bibliotecas 21). Si no indica ninguna biblioteca, la función aparecerá en una biblioteca predeterminada llamada `user`.
- *Sintaxis:* Campo opcional. Introduzca en formato texto una breve descripción de la sintaxis de la función (por ejemplo, los parámetros que se esperan). Este texto se mostrará junto a la función en la ventana **Bibliotecas** y no afecta a la implementación de la función.
- *Detalle:* Campo opcional. Introduzca en formato texto una descripción libre de la función. Este texto se mostrará cuando pase el cursor sobre la función en la ventana **Bibliotecas** o en otros contextos.
- *Uso inline:* Marque esta casilla si quiere crear la función como inline. Para más información, consulte el apartado *Funciones definidas por el usuario inline y regulares*.

3. Haga clic en **Aceptar**. La función aparecerá automáticamente en la biblioteca que haya indicado antes. Ahora también puede crear la nueva función en la ventana de asignaciones (esta es una

asignación independiente a la que se hace referencia como la *asignación de la función*). Como toda función necesita un componente de salida, la asignación de la función incluye uno por defecto.

4. Agregue a la asignación de la función todos los componentes requeridos en la definición de la función como haría para cualquier asignación estándar.

Para usar la función definida por el usuario en una asignación arrástrela desde la ventana **Bibliotecas** hasta el área principal de asignación. Consulte también *Llamar e importar funciones definidas por el usuario* más abajo.

Crear una función definida por el usuario a partir de componentes ya existentes
Para crear una función definida por el usuario a partir de componentes ya existentes siga estos pasos:

1. Seleccione varios componentes de la asignación dibujando un rectángulo con el cursor. También puede seleccionar varios componentes haciendo clic en cada uno de ellos mientras mantiene pulsada la tecla **Ctrl**.
2. Seleccione **Función | Crear una función definida por el usuario a partir de la selección**. También puede hacer clic en el botón de la barra de herramientas.
3. Siga los pasos 2-4 que hemos descrito en *Crear una función definida por el usuario desde cero*.

Funciones definidas por el usuario inline y regulares
Existen dos tipos de funciones definidas por el usuario: *inline* y de tipo *estándar*. Puede indicar si una función debe ser inline o de tipo estándar al crearla. Las funciones inline y de tipo estándar se comportan de forma diferente a la hora de generar código, en cuanto a recursividad y al tener varios parámetros de salida. En la tabla siguiente puede ver las diferencias principales entre estos dos tipos de funciones.

Funciones inline (borde discontinuo)	**Funciones regulares (borde sólido)**
El código de la función definida por el usuario se inserta en todos los lugares en los que se llama a la función, lo que aumenta considerablemente el tamaño del código.	El código de la función definida por el usuario se genera una vez y sus componentes de entrada se le pasan como valores de parámetros. Si se llama varias veces a la función, esta se evalúa cada vez con los valores de parámetro
Las funciones inline pueden tener varios componentes de salida y por tanto devolver varios valores.	Las funciones de tipo estándar sólo pueden tener un componente de salida. Para que devuelvan varios valores puede declarar que la salida es de tipo complejo (por ejemplo, estructura XML), lo que permitiría pasar varios valores al usuario.
No se puede llamar a las funciones inline de forma recursiva.	Se puede llamar de forma recursiva a las funciones de tipo estándar.
Las funciones inline no permiten establecer un contexto de prioridad 418 en un parámetro.	Las funciones estándar permiten establecer un contexto de prioridad en un parámetro.

Nota: cambiar el tipo de una función definida por el usuario de inline a estándar y viceversa puede afectar el contexto de la asignación 409, lo que a su vez puede afectar al resultado de la asignación.

Importar y llamar a funciones definidas por el usuario

Una vez haya creado una función definida por el usuario puede llamarla desde la asignación en que la creó o desde cualquier otra.

Llamar a una función definida por el usuario desde la misma asignación

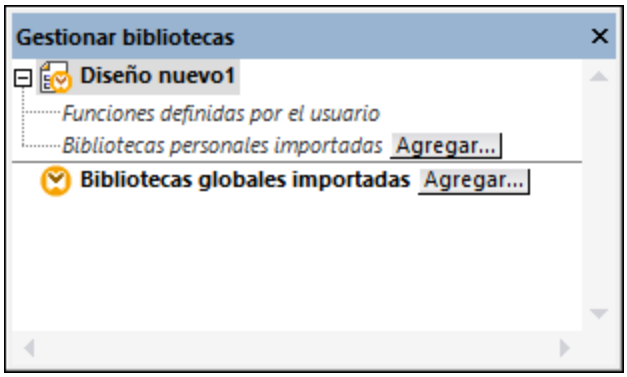
Para llamar a una función definida por el usuario desde la misma asignación siga estos pasos:

1. Busque la función en la ventana **Bibliotecas**, bajo la biblioteca que indicó al crear la función. Para ello comience a escribir su nombre en la ventana **Bibliotecas**.
2. Arrastre la función desde la ventana **Bibliotecas** hasta la asignación principal Ahora puede conectarla a todos los parámetros necesarios.

Importar una función definida por el usuario desde otra asignación

Para importar una función definida por el usuario desde otra asignación siga estos pasos:

1. Haga clic en el botón **Agregar o quitar bibliotecas**, en la parte inferior de la ventana Bibliotecas 21. Se abre la ventana **Gestionar bibliotecas**.



2. Para importar funciones como biblioteca local (dentro del archivo de asignación actual solamente), haga clic en **Agregar** bajo el nombre de la asignación activa. Para importar funciones como biblioteca global (a nivel de programa), haga clic en **Agregar**, junto a **Bibliotecas globales importadas**. Cuando se importa una biblioteca de forma local, puede hacer que la ruta de acceso al archivo de la biblioteca sea relativa al archivo de asignación. Con las bibliotecas globales la ruta siempre es absoluta.

3. Navegue hasta el archivo **`.mdf`** que contiene la función definida por el usuario y haga clic en **Abrir**. Un mensaje le informará de que se añadió una biblioteca nueva a la que puede acceder desde la ventana **Bibliotecas**.

Ahora puede arrastrar cualquiera de las funciones importadas en la asignación actual desde la ventana **Bibliotecas** hasta la asignación para usarlas. Para más información consulte Gestionar bibliotecas de funciones 200.

Asignaciones y credenciales (Enterprise Edition)

Si el archivo **`.mdf`** que se importa contiene credenciales, estas aparecen como importadas (con un fondo amarillo) en el Gestor de credenciales. Por defecto, las credenciales importadas no se guardan en la asignación, pero si quiere hacerlo basta con crear una copia local y guardar esas credenciales en la asignación principal.

Editar funciones definidas por el usuario

Para edita una función definida por el usuario siga estos pasos:

1. Abra la asignación que contiene la función definida por el usuario.
2. Haga doble clic en la barra del título de la función definida por el usuario en la asignación. La ventana Asignación cambia para mostrar el contenido de la función, en el que puede añadir, editar o eliminar componentes como desee.
3. Para cambiar las propiedades de la función (como el nombre o la descripción), haga clic con el botón derecho en un punto vacío de la asignación y seleccione **Configurar asignación** en el menú contextual. Otra opción es hacer clic en el botón de la barra de herramientas .

También puede editar una función haciendo doble clic en su nombre en la ventana **Bibliotecas**. Sin embargo, sólo se pueden abrir de esta forma las funciones que estén en el documento activo en ese momento. Si hace doble clic en una función definida por el usuario que se creó en otra asignación, se abrirá esa asignación en una ventana nueva. Si edita o elimina una función definida por el usuario que se ha importado a varias asignaciones, todas esas asignaciones se verán afectadas por el cambio.

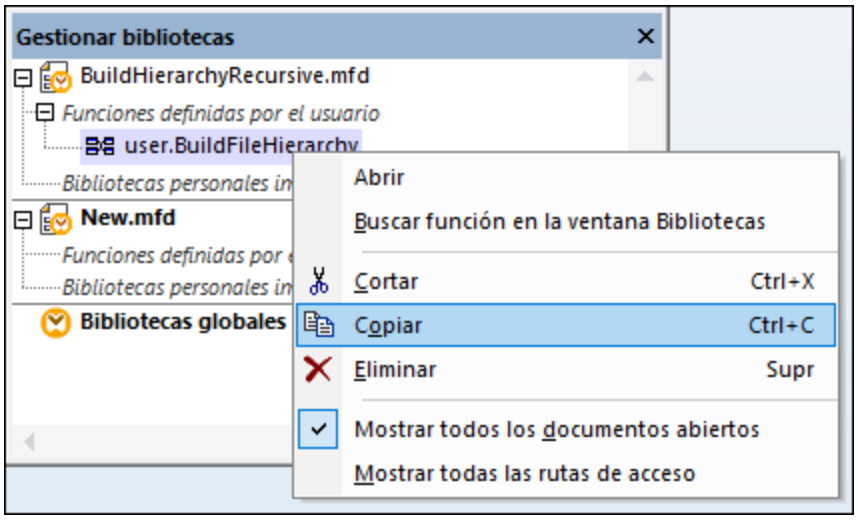
Para volver a la asignación principal haga clic en el botón , que está en la esquina superior izquierda de la ventana de la asignación.

Además, se conserva un historial mientras navega por las pestañas de MapForce, incluidas las funciones definidas por el usuario. Para volver a pestañas visitadas con anterioridad, haga clic en los botones de la barra de herramientas y . Los atajos de teclado correspondientes para estas acciones son respectivamente **Alt+flecha izquierda** y **Alt+flecha derecha**.

Copiar y pegar funciones definidas por el usuario

Para copiar una función definida por el usuario en otra asignación siga estos pasos:

1. Abra la ventana Gestionar bibliotecas 200.
2. Haga clic con el botón derecho en un área vacía de la ventana **Bibliotecas** y seleccione la opción **Mostrar todos los documentos abiertos**.
3. Abra las asignaciones de origen y de destino. Asegúrese de que ambas están guardadas en disco. De lo contrario las rutas de acceso no se pueden resolver correctamente. Consulte también Rutas relativas y las acciones cortar y pegar 42.
4. En la asignación de origen de la ventana **Gestionar bibliotecas** haga clic con el botón derecho en la función definida por el usuario en cuestión y seleccione **Copiar** en el menú contextual (*imagen siguiente*) o pulse **Ctrl+C**. Deje abierta la ventana **Gestionar bibliotecas**.



5. Cambie la asignación de destino (la ventana **Gestionar bibliotecas** cambiará a su vez), haga clic con el botón derecho en *Funciones definidas por el usuario* y seleccione **Pegar** en el menú contextual.

Eliminar funciones definidas por el usuario

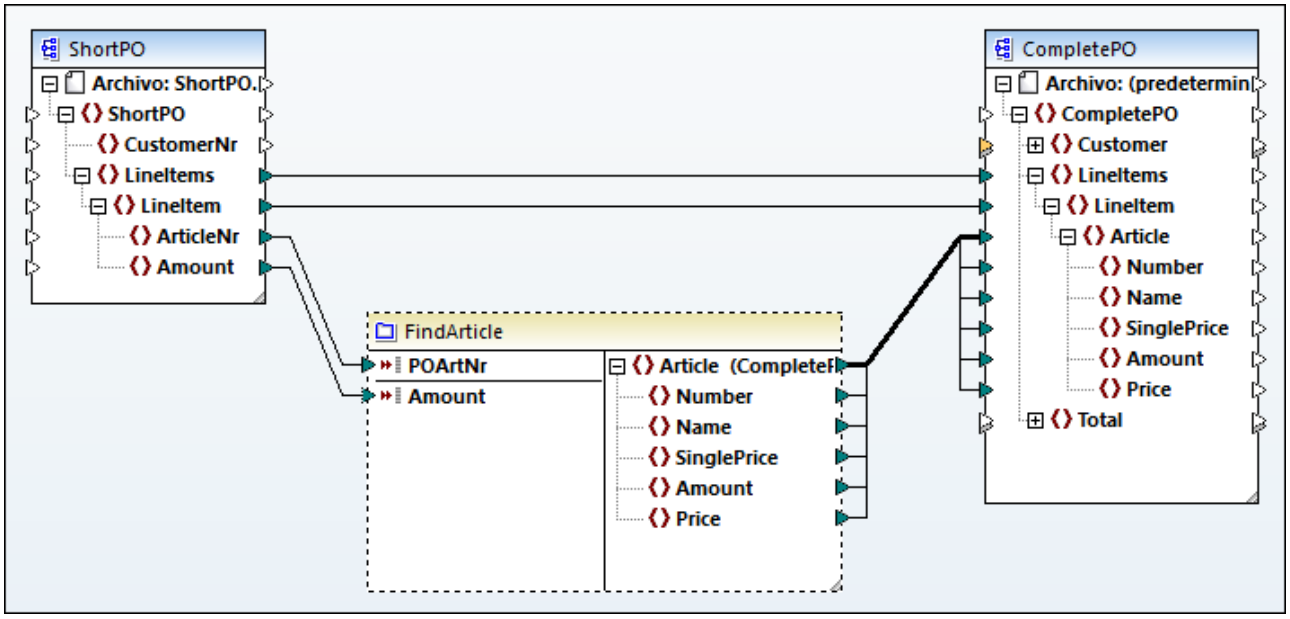
Para eliminar una función definida por el usuario siga estos pasos:

1. En la asignación haga doble clic en la barra del título de la función en cuestión.
2. Haga clic en el botón  , en la parte superior derecha de la ventana Asignación.
3. Si la función se usa en la asignación abierta actualmente, MapForce le preguntará si quiere reemplazar todas las instancias con componentes internos. Haga clic en **Sí** si quiere eliminar la función y reemplazar todas las instancias en que se llama a la función con componentes de la función. De esta forma la asignación sigue siendo válida aunque se haya eliminado la función. Sin embargo, si usa la función eliminada en una asignación externa, no será válida. Haga clic en **No** si quiere borrar la función y todos sus componentes de forma permanente. En este caso todas las asignaciones en las que se use esa función dejarán de ser válidas.

6.3.2 Parámetros en funciones definidas por el usuario

Cuando se crear una función definida por el usuario se debe indicar qué parámetros de entrada debe tomar (si los toma) y qué resultados debe devolver. Aunque los parámetros de entrada no siempre son necesarios, es obligatorio en todos los casos incluir un parámetro de salida. Los parámetros de las funciones pueden ser de tipo simple (como una `cadena` o un `entero`) o una estructura de tipo complejo 212. Por ejemplo, la función definida por el usuario `FindArticle` que ve más abajo tiene dos parámetros de entrada y uno de salida:

- `POArtNr` es un parámetro de entrada de tipo `cadena`.
- `Amount` es un parámetro de entrada de tipo `entero`.
- `CompletePO` es un parámetro de salida con una estructura XML compleja.



Orden de los parámetros
Cuando una función definida por el usuario tiene varios parámetros de entrada o salida, puede cambiar el orden en que aparecen cuando se llama a esa función. El orden de los parámetros en la asignación de la función (empezando por arriba) dicta el orden en que aparecen cuando se llama a la función.

Importante

- Los parámetros de entrada y de salida se ordenan según su posición de arriba a abajo. Por tanto, si mueve el parámetro `input3` a la parte superior de la asignación de la función, se convertirá en el primer parámetro de esta función.
- Si dos parámetros tienen la misma posición vertical, entonces tiene precedencia el último.
- En el caso improbable de que varios parámetros tengan la misma posición, entonces se usa automáticamente un ID de componente interno.

Estructuras de tipo complejo

En la lista siguiente puede ver las estructuras en que se puede basar un parámetro de una función definida por el usuario.

MapForce Basic Edition
- Estructura de XML Schema

MapForce Professional Edition
- Estructura de XML Schema
- Estructura de BD

MapForce Enterprise Edition
- Estructura de XML Schema
- Estructura de BD
- Estructura EDI
- Estructura FlexText
- Estructura de JSON Schema

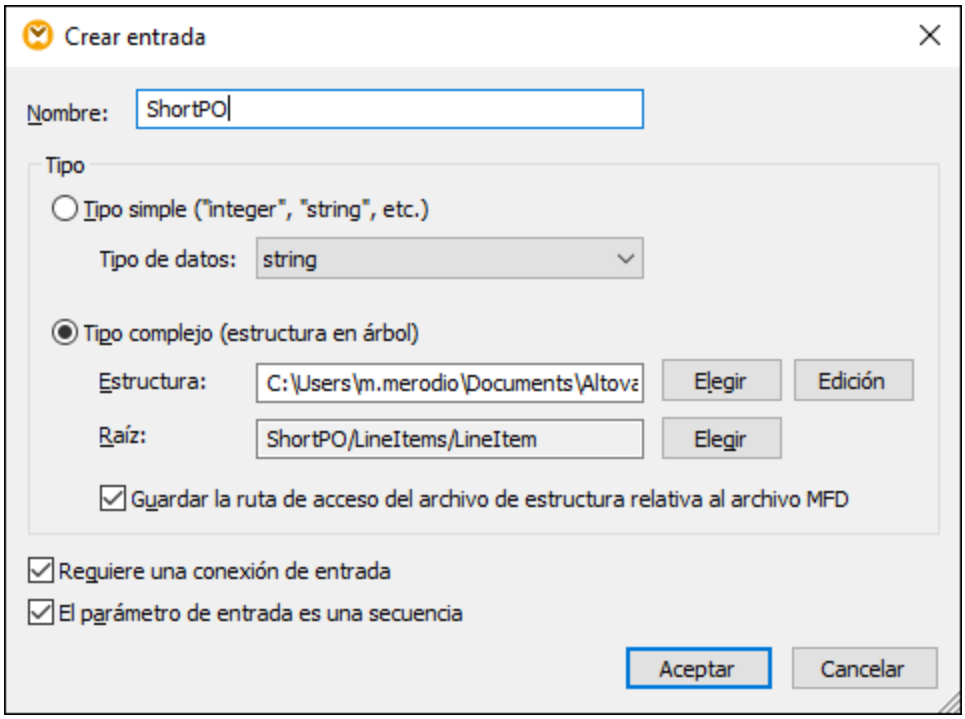
Las funciones definidas por el usuario se basan en estructuras de base de datos (en las ediciones Professional y Enterprise)

MapForce permite crear parámetros en funciones definidas por el usuario basadas en bases de datos con una estructura de tablas relacionadas. La estructura de tablas relacionadas representa una estructura en memoria que no tiene conexión con la BD en tiempo de ejecución. Esto también significa que no hay una forma automática de administrar las claves foráneas ni acciones de tabla en los parámetros o las variables.

Agregar parámetros

Para agregar un parámetro de entrada o de salida siga estos pasos:

1. Cree una función definida por el usuario 206 o abra una que ya exista 209.
2. Ejecutar el comando de menú **Función | Insertar componente de entrada** o **Función | Insertar componente de salida** (*imagen siguiente*). También puede hacer clic en (**Insertar componente de entrada**) or (**Insertar componente de salida**) en la barra de herramientas.



3. Elija si los componentes de entrada o de salida deben ser de tipo simple o complejo (*véase el cuadro de diálogo anterior*). Consulte la lista de estructuras complejas disponibles más arriba. Por ejemplo, para crear un parámetro XML de tipo complejo haga clic en **Elegir**, junto a *Estructura*, y navegue hasta el esquema XML que describe la estructura requerida.

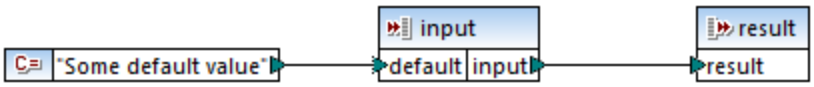
Si la asignación de la función ya incluye esquemas XML, estos pasan a estar disponibles para seleccionarlos como estructuras. De lo contrario puede seleccionar un esquema nuevo para que proporcione estructura al parámetro. Los mismo ocurre con bases de datos y otras estructuras complejas, siempre que sean compatibles con su versión de MapForce. Con las estructuras XML es posible seleccionar el elemento raíz pata la estructura si el esquema XML lo permite. Para indicar el elemento raíz haga clic en **Elegir**, junto a *Raíz*, y seleccione el elemento raíz en el cuadro de diálogo que se abre.

Si la casilla *Guardar ruta de acceso de la estructura como relativas al archivo MFD* está marcada, la ruta absoluta de la estructura pasará a ser relativa a la asignación actual cuando la guarde. Para más información

consulte el apartado relativa y ruta absoluta 42. Las casillas *Requiere una conexión de entrada* y *El parámetro de entrada es una secuencia* se explican en los siguientes apartados.

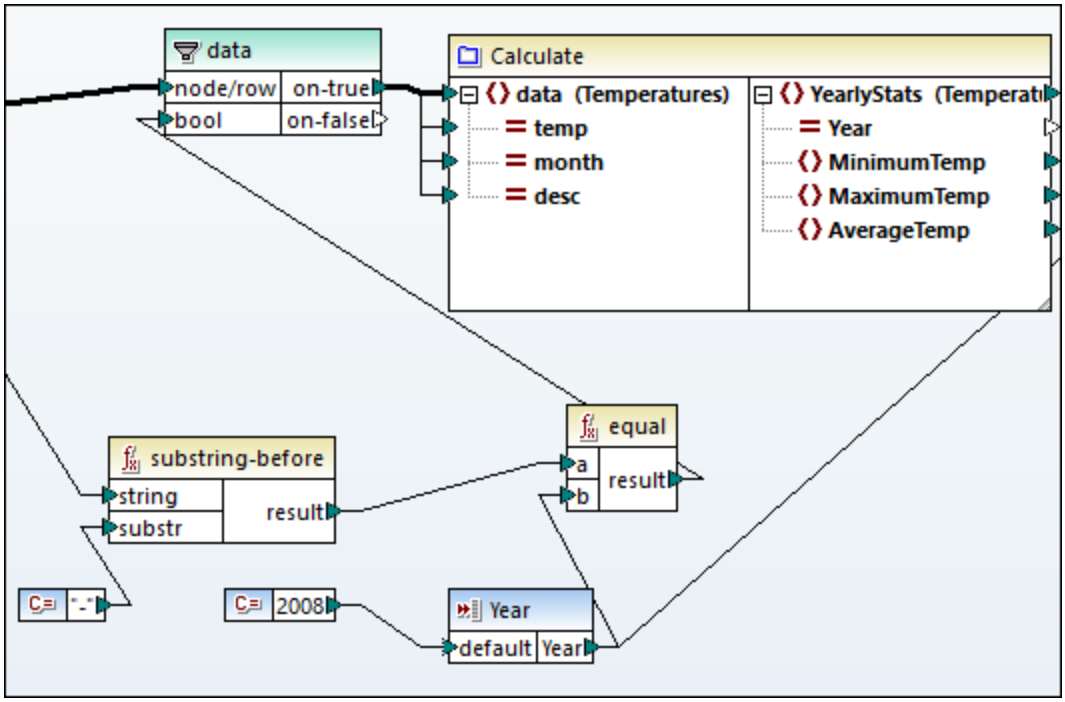
Requiere una conexión de entrada
Para hacer que un parámetro sea obligatorio en una función definida por el usuario, marque la casilla *Requiere una conexión de entrada* (*véase el cuadro de diálogo de más arriba*). Si desmarca esta casilla, el parámetro se convierte en opcional y aparece con un borde discontinuo en la asignación.

También puede indicar un valor de parámetro predeterminado si lo conecta a la entrada `default` de un parámetro (*ejemplo siguiente*). El valor predeterminado sólo se aplica si no hay ningún otro valor. Si el parámetro opcional recibe un valor cuando se llama a la función, ese valor tiene prioridad sobre el valor predeterminado.

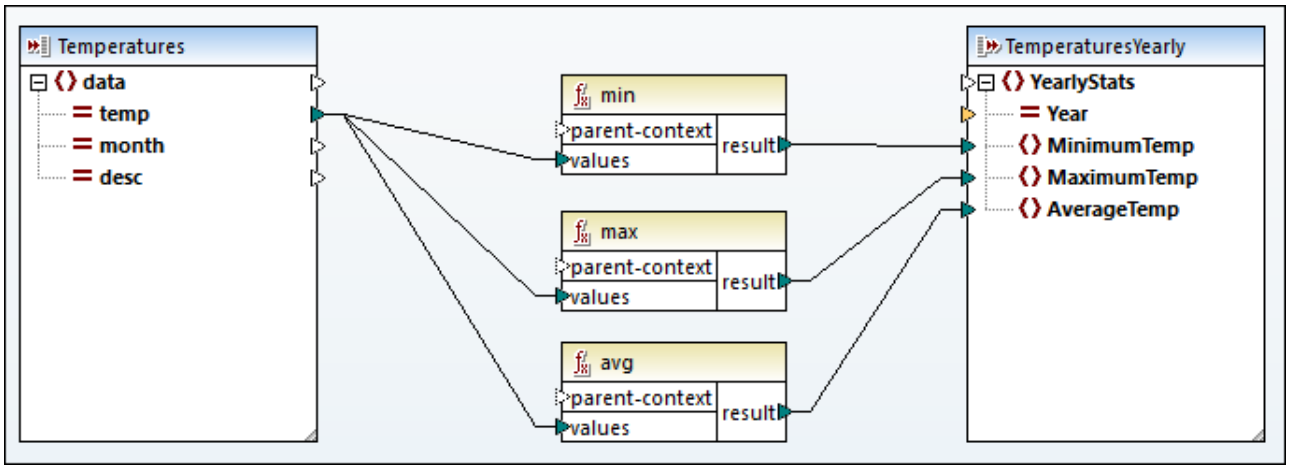


El componente de entrada es una secuencia
Puede decidir si un parámetro de una función se debe tratar como un valor único (opción predeterminada) o como una secuencia. Una secuencia es un rango de cero o más valores. Una secuencia puede ser útil si la función definida por el usuario espera que los datos de entrada sean una secuencia para calcular los valores de esa secuencia, por ejemplo, llamando a funciones como **`avg`**, **`min`**, **`max`**. Para tratar el componente de entrada del parámetro como una secuencia, marque la casilla *El parámetro de entrada es una secuencia*. Recuerde que esta casilla sólo se habilita para funciones definidas por el usuario estándar 208.

En la asignación siguiente puede ver islustrado el uso de una secuencia: **`MapForceExamples\InputIsSequence.mfd`**. En el extracto de esta asignación de elementos (*imagen siguiente*) el filtros de `datos` está conectado a la función definida por el usuario `Calculate`. El resultado del filtro es una secuencia de elementos. Por tanto, el componente de entrada de la función se definió como secuencia.



A continuación puede ver la implementación de la función `Calculate` que suma todos los valores de la secuencia: Las funciones **`avg`**, **`min`** y **`max`** se ejecutan en la secuencia de entrada. Para ver la estructura interna de la función `Calculate` hace doble clic en el encabezado del componente `Calculate` de la asignación anterior.



Por lo general, los datos de entrada (sean una secuencia o no) determinan cómo de a menudo se llama a la función:

- Cuando los datos de entrada se conectan al parámetro *sequence*, se llama a la función definida por el usuario *una sola vez* y la secuencia completa se pasa a la función definida por el usuario.
- Cuando los datos de entrada se conectan al parámetro *non-sequence*, se llama a la función definida por el usuario *una vez por cada elemento* de la secuencia.

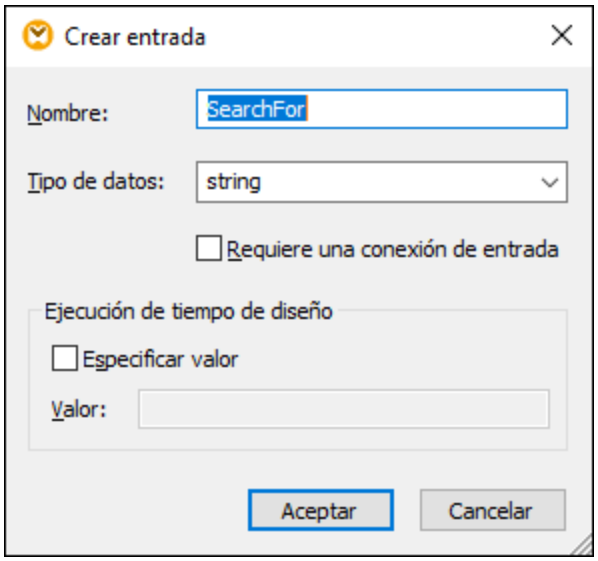
- Si conecta una secuencia vacía a un parámetro que no sea una secuencia, no se llama a la función. Esto puede ocurrir si la estructura de origen tiene elementos opcionales o si una de las condiciones de un filtro no devuelve coincidencias. Para evitarlo, use la función `substitute-missing` 307 antes de la entrada de la función para asegurarse de que la secuencia nunca esté vacía. También puede definir el parámetro como secuencia y agregar controles para la secuencia vacía dentro de la función.

Es posible que también necesite la casilla *El componente de salida es una secuencia* para los parámetros de salida. Cuando una función pasa una secuencia de varios valores a su componente de salida y este no se ha definido como secuencia, la función devuelve solamente el primer elemento de la secuencia.

6.3.3 Búsqueda recursiva

En esta sección explicamos cómo realizar búsquedas recursivas de datos en un archivo XML de origen con ayuda de las funciones definidas por el usuario. Para probar la búsqueda recursiva de la función definida por el usuario necesita esta asignación: **`MapForceExamples\RecursiveDirectoryFilter.mfd`**. En la asignación siguiente la función definida por el usuario `FilterDirectory` recibe datos del archivo de origen `Directory.xml` y del componente de entrada simple `SearchFor` que suministra la extensión `.xml`. Una vez la función ha procesado los datos, se asigna al archivo de destino.

La asignación principal (*imagen siguiente*) describe el diseño general de la asignación. La forma en que la función definida por el usuario procesa los datos se define aparte, en la asignación de la función (consulte más abajo *Implementación de de funciones definidas por el usuario*).



Objetivo
Nuestro objetivo es hacer una lista de los archivos con la extensión **`.xml`** en los resultados mientras conservamos la toda la estructura del directorio. A continuación explicamos en detalle el proceso de asignación.

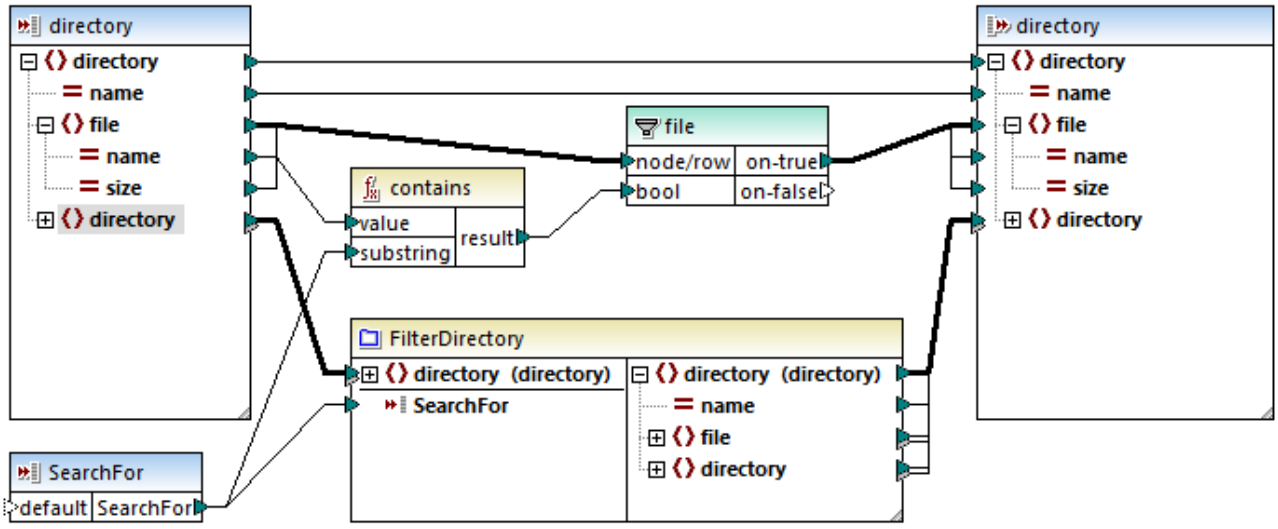
Archivo de origen

Más abajo puede ver un extracto del archivo XML de origen (**`Directory.xml`**) que contiene información sobre los archivos y directorios. Observe que los archivos de esta lista tienen extensiones distintas (e.g., **`.xml`**, **`.dtd`**, **`.sps`**). Según el esquema (**`Directory.xsd`**), el elemento `directory` puede tener descendientes `file` y `directory`. Todos los elementos `directory` son recursivos.

```
<directory name="ExampleSite">
  <file name="blocks.sps" size="7473"/>
  <file name="blocks.xml" size="670"/>
  <file name="block_file.xml" size="992"/>
  <file name="block_schema.xml" size="1170"/>
  <file name="contact.xml" size="453"/>
  <file name="dictionaries.xml" size="206"/>
  <file name="examplesite.dtd" size="230"/>
  <file name="examplesite.spp" size="1270"/>
  <file name="examplesite.sps" size="20968"/>
  ...
  <directory name="output">
     <file name="examplesite1.css" size="3174"/>
     <directory name="images">
        <file name="blank.gif" size="88"/>
        <file name="block_file.gif" size="13179"/>
        <file name="block_schema.gif" size="9211"/>
        <file name="nav_file.gif" size="60868"/>
        <file name="nav_schema.gif" size="6002"/>
     </directory>
  </directory>
</directory>
```

Implementación de funciones definidas por el usuario

Para ver la implementación interna de la función definida por el usuario haga doble clic en su encabezado en la asignación principal. La función definida por el usuario es recursiva, es decir, se llama a sí misma. Al estar conectada al elemento recursivo `directory`, se llama a esta función tantas veces como elementos anidados `directory` haya en la instancia XML de origen. Para que admita llamadas recursivas, la función debe ser estándar[208].



La implementación de la función definida por el usuario consiste en dos partes: (i) definir los archivos y (ii) definir el directorio en que se realiza la búsqueda.

Definir archivos

Así es como la función definida por el usuario procesa los archivos: La función **`contains`** comprueba si la primera cadena (el nombre del archivo) contiene la subcadena `.xml` (dada por el componente de entrada simple `SearchFor`). Si la función devuelve `true`, se escribe en el resultado el nombre del archivo con la extensión `.xml`.

Procesar directorios descendientes

Los directorios descendientes del directorio actual se envían como entrada a la función definida por el usuario actual. Por tanto, esa función va comprobando si existen archivos con la extensión **`.xml`** en todos los elementos `directory`.

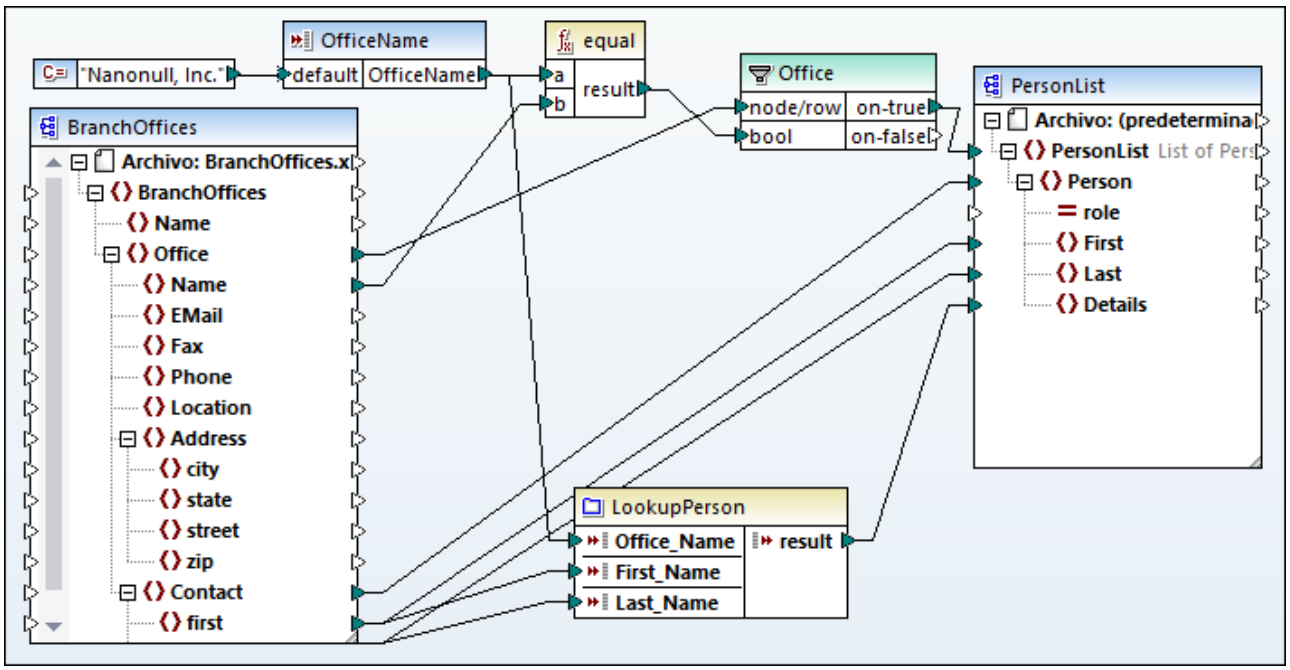
Resultados

Al hacer clic en el panel **Resultados**, MapForce muestra solamente los archivos que tienen la extensión **`.xml`** (*véase el extracto más abajo*).

```
<directory name="ExampleSite">
  <file name="blocks.xml" size="670"/>
  <file name="block_file.xml" size="992"/>
  <file name="block_schema.xml" size="1170"/>
  <file name="contact.xml" size="453"/>
  ...
  <directory name="output">
     <directory name="images"/>
  </directory>
</directory>
```


6.3.4 Implementación de la búsqueda

En este apartado explicamos cómo buscar información sobre los empleados y presentar esa información de forma adecuada. Para probar la implementación de la búsqueda necesitará esta asignación: **`MapForceExamples\PersonListByBranchOffice.mfd`**.



Objetivos
En este caso los objetivos son:

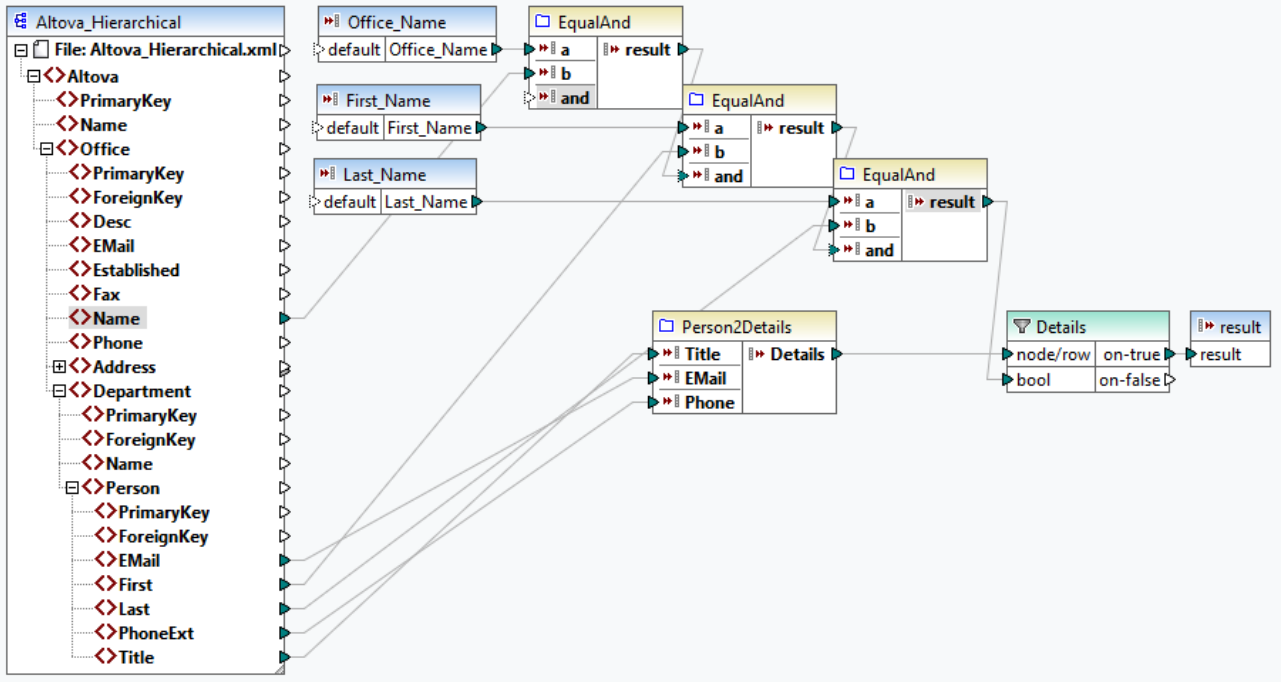
- Buscar información sobre cada uno de los empleados (su extensión de teléfono, dirección de correo electrónico y título) en un archivo XML aparte.
- Presentar estos datos como lista separada por comas y asignarla al elemento `Details` del archivo XML de destino.
- Extraer información sobre los empleados de una sola filial llamada Nanonull, Inc.

Para ello hemos diseñado la asignación de la siguiente forma:

- Para visualizar solamente los empleados de Nanonull, Inc. la asignación usa el filtro `Office`.
- Para buscar información sobre los empleados en un archivo XSLT diferente la asignación llama a la función definida por el usuario `LookupPerson`. La implementación de esta función definida por el usuario se describe más abajo.
- Para procesar la información de los empleados, la función `LookupPerson` llama internamente a otras funciones que obtienen y concatenan la información de los empleados. Todas estas operaciones están dentro de la asignación de la función y no son visibles en la asignación principal. Después, la función `LookupPerson` asigna los datos del empleado al elemento `Details` en `PersonList`.

Implementación de LookupPerson

La funcionalidad de búsqueda la proporciona la función `LookupPerson`, sobre la que encontrará más información más abajo. Para ver la implementación interna de la función definida por el usuario haga doble clic en su encabezado en la asignación principal.

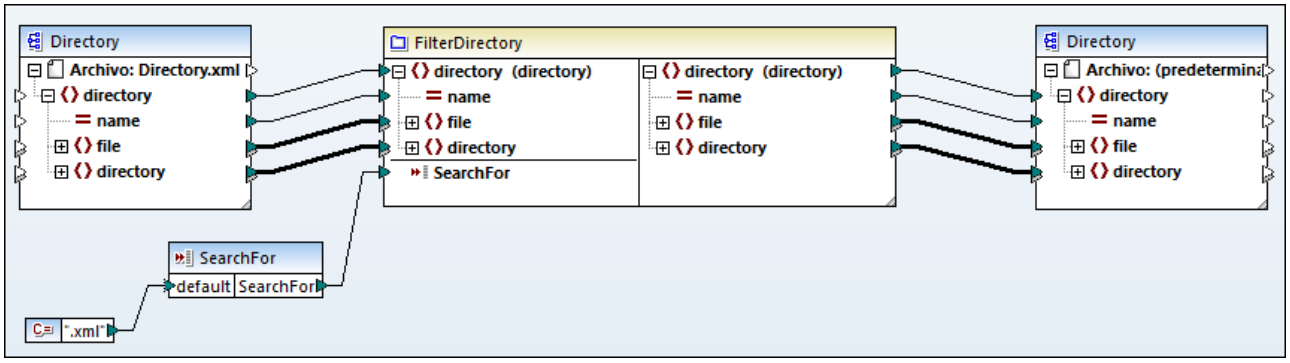


Así se configuró la unción definida por el usuario:

- Los datos se obtienen del archivo XML **`Altova_Hierarchical.xml`**: (i) el nombre de la oficina y el nombre y apellido de los empleados, que se usan para seleccionar solamente empleados de Nanonull, Inc., y (ii) el correo electrónico, el título y la extensión de teléfono, que se concatenan en una sola cadena. A continuación describimos las definiciones de las funciones `EqualAnd` y `Person2Detail`.
- La función definida por el usuario también tiene tres parámetros de entrada que proporcionan los valores de búsqueda `Office_Name`, `First_Name` y `Last_Name`. El valor del parámetro `Office_Name` se obtiene de la entrada `OfficeName` de la asignación principal, y los valores de `First_Name` y `Last_Name` los da el componente `BranchOffices` de la asignación principal.
- El valor de la función `EqaulAnd` (`true` o `false`) se pasa al filtro `Details` cada vez que se procesa la información (título, correo electrónico, teléfono) de un empleado. Cuando el filtro `Details` obtiene el valor `true` se entiende que la búsqueda se ha realizado correctamente y que se pueden pasar los datos obtenidos a la asignación principal. De lo contrario se examina el elemento siguiente en el contexto hasta que termine el bucle.

Implementación de EqualAnd

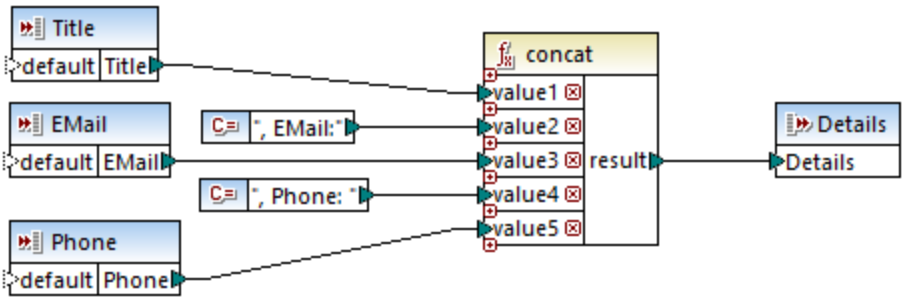
La función `EqualAnd` (*véase más abajo*) es una función definida por el usuario aparte que está definida dentro de la función definida por el usuario `LookupPerson`. Para ver la estructura interna de `EqualAnd` haga doble clic en su encabezado.



`EqualAnd` primero comprueba si a es igual a b; si el resultado es `true`, este se pasa al primer parámetro de la función **`logical-and`**. Si los dos valores son true en la función **`logical-and`**, el resultado también es true y se pasa a la siguiente función `EqualAnd`. El resultado de la tercera función `EqualAnd` (*véase LookupPerson más arriba*) se pasa al filtro `Details`.

Implementación de Person2Detail

La función definida por el usuario `Person2Details` es otra función que se encuentra dentro de `LookupPerson`. Esta función (*véase más abajo*) concatena tres valores (obtenidos de **`Altova_Hierarchical.xml`**) y dos constantes de texto.



Resultados

Al hacer clic en el panel **Resultados**, MapForce muestra la información recuperada de los empleados de Nanonull Inc (*véase a continuación*).

```
<PersonList>
  <Person>
     <First>Vernon</First>
     <Last>Callaby</Last>
     <Details>Office Manager, EMail:v.callaby@nanonull.com, Phone: 582</Details>
  </Person>
  <Person>
     <First>Frank</First>
     <Last>Further</Last>
     <Details>Accounts Receivable, EMail:f.further@nanonull.com, Phone: 471</Details>
  </Person>
  ...
</PersonList>
```

6.4 Funciones personales

En esta sección explicamos cómo importar funciones personales XSLT[223].

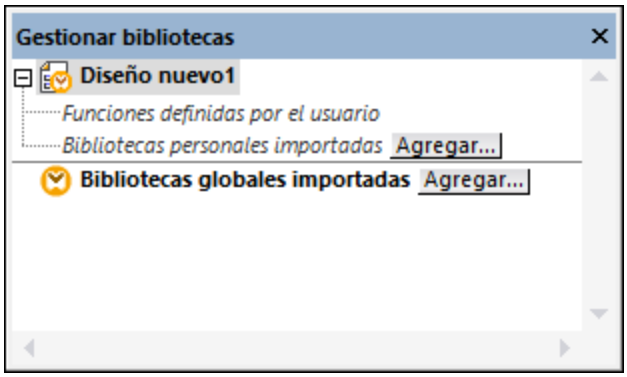
6.4.1 Importar funciones XSLT 1.0/2.0 personales

Puede ampliar las bibliotecas de funciones XSLT 1.0, 2.0 y 3.0 de MapForce con sus propias funciones personales, siempre y cuando estas devuelvan tipos simples.

> Solamente se admiten funciones personales que devuelvan tipos simples (p. ej. cadenas).

Para importar funciones de un archivo XSLT:

1. Haga clic en el botón **Agregar o quitar bibliotecas**, en la parte inferior de la ventana Bibliotecas[21]. Se abre la ventana **Gestionar bibliotecas**.



2. Para importar funciones como biblioteca local (dentro del archivo de asignación actual solamente), haga clic en **Agregar** bajo el nombre de la asignación activa. Para importar funciones como biblioteca global (a nivel de programa), haga clic en **Agregar**, junto a **Bibliotecas globales importadas**. Cuando se importa una biblioteca de forma local, puede hacer que la ruta de acceso al archivo de la biblioteca sea relativa al archivo de asignación. Con las bibliotecas globales la ruta siempre es absoluta.

3. Navegue hasta el archivo .xsl que contiene las funciones y haga clic en **Abrir**. Aparece un cuadro de mensaje que le informa de que se ha añadido una biblioteca nueva.

Los archivos XSLT importados aparecen como bibliotecas en la ventana Bibliotecas y muestran todas las plantillas con nombres como si fueran funciones. Si no puede ver la biblioteca importada, compruebe que el lenguaje de transformación[17] seleccionado es XSLT (*véase también* Gestionar bibliotecas de funciones[200]).

Importante:

- Para poder importarlas en MapForce las funciones deben declararse en el archivo XSLT como plantillas con nombre según lo estipulado por la especificación XSLT. También puede importar

funciones que aparezcan en un documento XSLT 2.0 con el formato `<xsl:function name="MiFunción">`. Si el archivo XSLT importado importa o incluye otros archivos XSLT, también se importarán dichos archivos y sus funciones.

- Los conectores de entrada asignables de las funciones importadas personales dependen del número de parámetros utilizado en la llamada a plantilla. También se admiten parámetros opcionales.
- No se admiten espacios de nombres.
- Si realiza cambios en archivos XSLT que ya están importados en MapForce, los cambios se detectan automáticamente y puede elegir si los archivos se vuelven a cargar.
- Cuando escriba plantillas con nombre, asegúrese de que las instrucciones XPath utilizadas en la plantilla están enlazadas al espacio de nombres correcto. Para ver los enlaces de espacio de nombres de la asignación genere una vista previa del código XSLT de salida [66].

Tipos de datos en XPath 2.0

Si el documento XML hace referencia a un esquema XML y es válido según este esquema, deberá convertir o construir explícitamente tipos de datos que no se conviertan implícitamente al tipo de datos necesario por medio de una operación.

En el modelo de datos de XPath 2.0 que utiliza el motor XSLT 2.0 de Altova todos los valores de nodos **atomizados** del documento XML tienen asignado el tipo de datos `xs:untypedAtomic`. El tipo `xs:untypedAtomic` es adecuado para las conversiones de tipo implícitas.

Por ejemplo:

- la expresión `xs:untypedAtomic("1") + 1` da como resultado un valor de 2 porque el valor `xdt:untypedAtomic` se convierte **implícitamente** en `xs:double` por medio del operador de suma.
- los operadores aritméticos convierten los operandos implícitamente en `xs:double`.
- los operadores de comparación de valores convierten los operandos en `xs:string` antes de compararlos.

Temas relacionados:

Ejemplo: agregar funciones XSLT 1.0 personales [224]
Ejemplo: sumar valores de nodos [227]
Implementación del motor XSLT 1.0 [489]
Implementación del motor XSLT 2.0 [490]

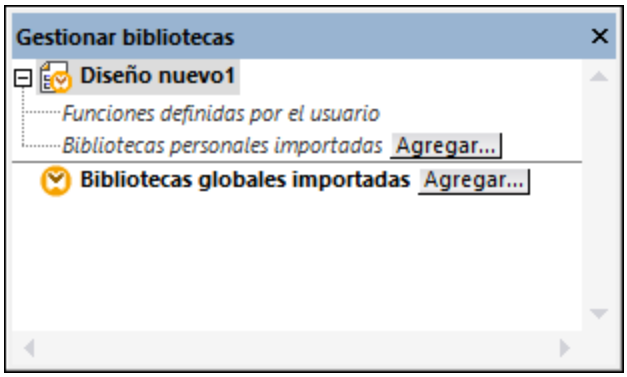
6.4.1.1 Ejemplo: agregar funciones XSLT personales

Este ejemplo explica cómo importar funciones XSLT 1.0 personales en MapForce. Los archivos necesarios para seguir este ejemplo están en la carpeta **`C:\Usuarios\<usuario>\Documentos\Altova\MapForce2024\MapForceExamples`**.

- **Name-splitter.xslt:** este archivo XSLT define una plantilla con nombre llamada "**tokenize**" que tiene un solo parámetro llamado "string". La plantilla recorre una cadena de entrada y separa todas las letras mayúsculas de la cadena con un espacio.
- **Name-splitter.xml:** el archivo de instancia XML de origen que se debe procesar.
- **Customers.xsd:** el esquema XML de origen.
- **CompletePO.xsd:** el esquema XML de destino.

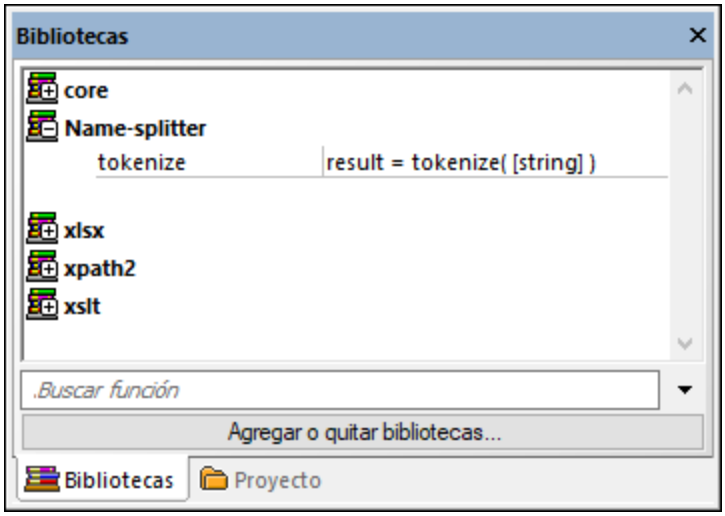
Para agregar una función XSLT personal:

1. Haga clic en el botón **Agregar o quitar bibliotecas**, en la parte inferior de la ventana Bibliotecas 21. Se abre la ventana **Gestionar bibliotecas**.



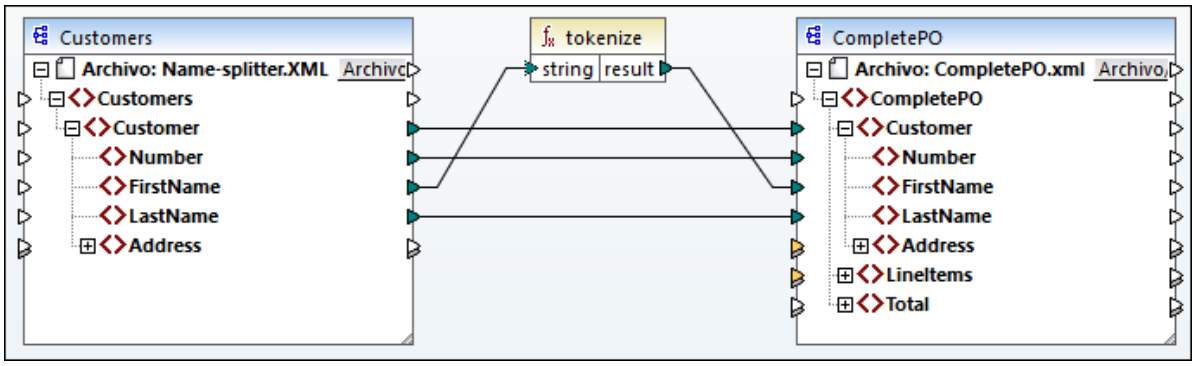
2. Para importar funciones como biblioteca local (dentro del archivo de asignación actual solamente), haga clic en **Agregar** bajo el nombre de la asignación activa. Para importar funciones como biblioteca global (a nivel de programa), haga clic en **Agregar**, junto a **Bibliotecas globales importadas**. Cuando se importa una biblioteca de forma local, puede hacer que la ruta de acceso al archivo de la biblioteca sea relativa al archivo de asignación. Con las bibliotecas globales la ruta siempre es absoluta.

3. Ahora haga clic en el botón **Agregar** y navegue hasta el archivo XSL o XSLT que contiene la plantilla con nombre que desea usar como función (en este caso **Name-splitter.xslt**). Haga clic en **Aceptar**. El nombre del archivo XSLT aparece en la ventana Bibliotecas junto con las funciones definidas como plantillas con nombre (en este ejemplo es **Name-splitter** con la función `tokenize`).



Para usar la función XSLT en la asignación de datos:

1. Arrastre la función **tokenize** hasta el área de asignación y cree las asignaciones de datos que aparecen en esta imagen:



2. Haga clic en el panel *XSLT* para ver el código XSLT de salida.

```
1   <?xml version="1.0" encoding="UTF-8"?>
2   <!-- ... -->
11  <xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:xs="http
12      <xsl:include href="file:///C:/Users/altova/Documents/Altova/MapForce2020/MapForceExamples
13      <xsl:output method="xml" encoding="UTF-8" byte-order-mark="no" indent="yes"/>
14      <xsl:template match="/">
15          <CompletePO>
16              <xsl:attribute name="xsi:noNamespaceSchemaLocation" namespace="http://www.w3.org/
    CompletePO.xsd'"/>
17              <xsl:for-each select="Customers/Customer">
18                  <Customer>
19                      <Number>
20                          <xsl:sequence select="xs:string(xs:integer(fn:string(Number)))"/>
21                      </Number>
22                      <FirstName>
23                          <xsl:call-template name="tokenize">
24                              <xsl:with-param name="string" select="FirstName" as="item()"/>
25                          </xsl:call-template>
26                      </FirstName>
27                      <LastName>
28                          <xsl:sequence select="fn:string(LastName)"/>
29                      </LastName>
30                  </Customer>
31              </xsl:for-each>
32          </CompletePO>
33      </xsl:template>
34  </xsl:stylesheet>
35
```

Nota: en cuanto una plantilla con nombre se utiliza en una asignación de datos, el archivo XSLT que contiene la plantilla con nombre **se incluye** en el código XSLT de salida (**xsl:include href**...) y **se le llama** con el comando **xsl:call-template**.

3. Haga clic en el panel *Resultados* para ver el resultado de la asignación de datos.

Para quitar bibliotecas XSLT personales de MapForce:

1. Haga clic en el botón **Agregar o quitar bibliotecas** situado en la parte inferior de la ventana Bibliotecas.
2. Haga clic en la biblioteca XSLT que desea eliminar y después haga clic en **Eliminar biblioteca** .

6.4.1.2 Ejemplo: sumar valores de nodos

Este ejemplo explica cómo procesar varios nodos de un documento XML y asignar el resultado en forma de un solo valor a un documento XML de destino. Concretamente el objetivo de esta asignación de datos es calcular el precio de todos los productos de un archivo XML de origen y escribirlo como un solo valor en un archivo XML de salida. Los archivos utilizados en este ejemplo están en la carpeta **<Documentos>\Altova\MapForce2024\MapForceExamples\Tutorial**:

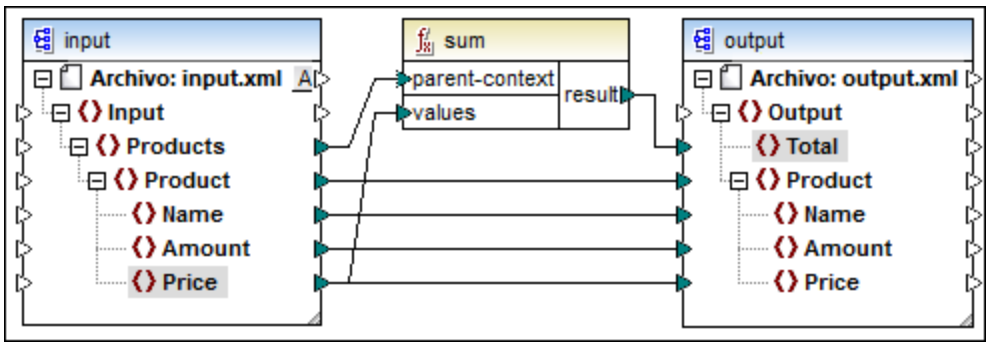
- **Summing-nodes.mfd:** el archivo de asignación de datos
- **input.xml:** el archivo XML de origen
- **input.xsd:** el esquema XML de origen
- **output.xsd:** el esquema XML de destino
- **Summing-nodes.xslt:** una hoja de estilos XSLT personal que contiene una plantilla con nombre que sumará los nodos.

Hay dos maneras de conseguir nuestro objetivo:

- con ayuda de la función de agregado **sum** 243 de la biblioteca **core**.
- importando una hoja de estilos XSLT personal en MapForce.

Solución nº1: usar la función de agregado "sum"

Para usar la función de agregado **sum** a la asignación basta con arrastrarla desde la ventana Bibliotecas hasta el panel de asignación. Recuerde que las funciones que aparecen en la ventana Bibliotecas dependen del lenguaje de transformación que esté seleccionado. Tras colocar la función en el área de asignación deberá crear las conexiones de asignación que aparecen en esta imagen:



Para más información sobre las funciones de agregado de la biblioteca **core** consulte el apartado core | aggregate functions (agregado) 237.

Solución nº2: usar una hoja de estilos XSLT personal

El objetivo del ejemplo es sumar los campos `Price` de los productos del archivo XML de origen (en este caso los productos A y B).

```
<?xml version="1.0" encoding="UTF-8"?>
<Input xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="input.xsd">
   <Products>
      <Product>
         <Name>ProductA</Name>
         <Amount>10</Amount>
         <Price>5</Price>
      </Product>
      <Product>
         <Name>ProductB</Name>
         <Amount>5</Amount>
         <Price>20</Price>
      </Product>
   </Products>
</Input>
```

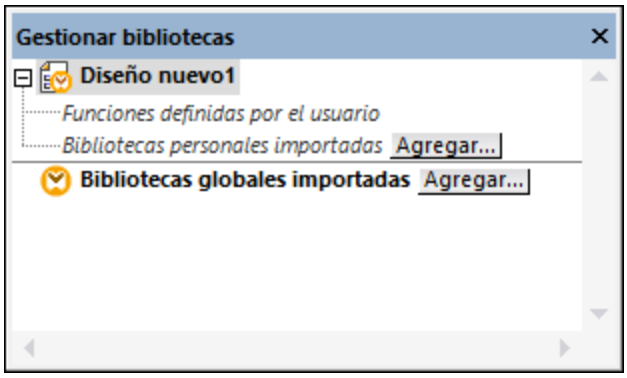
A continuación puede ver una hoja de estilos XSLT personal que usa la plantilla con nombre "Total" y un solo parámetro `string`. La plantilla recorre el archivo XML de entrada y suma todos los valores que obtiene la expresión `/Product/Price`.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>

   <xsl:template match="*">
      <xsl:for-each select=".">
      <xsl:call-template name="Total">
         <xsl:with-param name="string" select="."/>
      </xsl:call-template>
      </xsl:for-each>
   </xsl:template>

   <xsl:template name="Total">
   <xsl:param name="string"/>
      <xsl:value-of select="sum($string/Product/Price)"/>
   </xsl:template>
</xsl:stylesheet>
```

Nota: para sumar los nodos en XSLT 2.0 debe cambiar la declaración de hoja de estilos por `version="2.0"`.

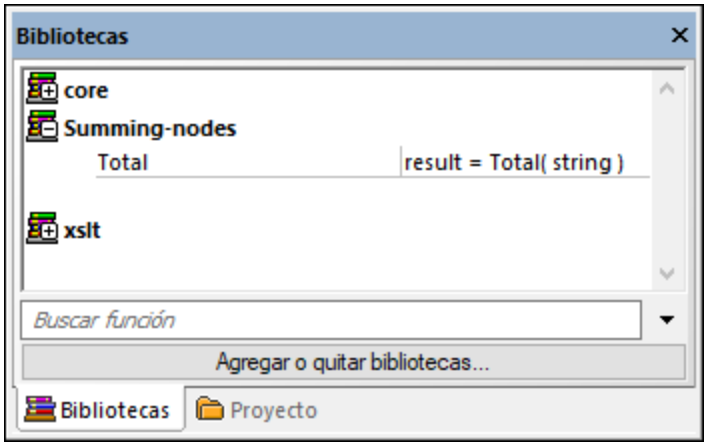
Para importar la hoja de estilos XSLT en MapForce:

1. Haga clic en el botón **Agregar o quitar bibliotecas**, en la parte inferior de la ventana Bibliotecas[21]. Se abre la ventana **Gestionar bibliotecas**.

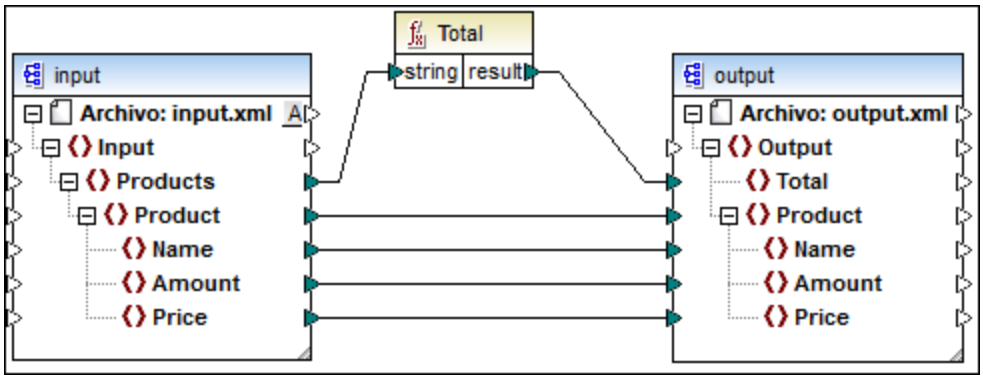


2. Para importar funciones como biblioteca local (dentro del archivo de asignación actual solamente), haga clic en **Agregar** bajo el nombre de la asignación activa. Para importar funciones como biblioteca global (a nivel de programa), haga clic en **Agregar**, junto a **Bibliotecas globales importadas**. Cuando se importa una biblioteca de forma local, puede hacer que la ruta de acceso al archivo de la biblioteca sea relativa al archivo de asignación. Con las bibliotecas globales la ruta siempre es absoluta.

3. En el cuadro de diálogo "Opciones" haga clic en el botón **Agregar**. Navegue hasta el archivo **<Documentos>\Altova\MapForce2024\MapForceExamples\Tutorial\Summing-nodes.xslt**.



4. Arrastre la función `Total` de la biblioteca recién añadida (**Summing-nodes**) hasta el área de asignación y cree las conexiones que aparecen en esta imagen:



Haga clic en el panel *Resultados* para obtener una vista previa del resultado de la asignación. La suma de los dos campos `Price` aparecen ahora en el campo `Total`.

```
<?xml version="1.0" encoding="UTF-8"?>
<Output xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="output.xsd">
   <Total>25</Total>
   <Product>
      <Name>ProductA</Name>
      <Amount>10</Amount>
      <Price>5</Price>
   </Product>
   <Product>
      <Name>ProductB</Name>
      <Amount>5</Amount>
      <Price>20</Price>
   </Product>
</Output>
```

6.5 Expresiones regulares

Al diseñar una asignación en MapForce puede usar expresiones regulares ("regex") en estos contextos:

- En el parámetro **pattern** de la función **tokenize-regexp** 318.

La sintaxis de las expresiones regulares en XSLT y XQuery se explican en Appendix F of "XML Schema Part 2: Datatypes Second Edition".

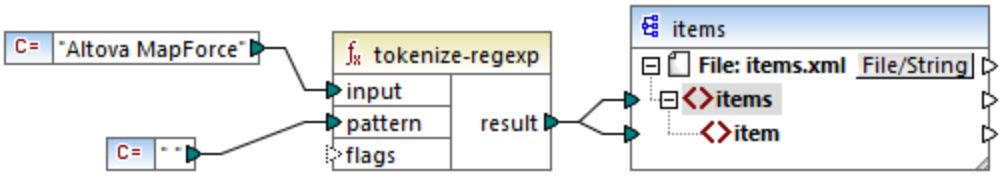
Nota: al generar código C++, C# o Java, las características avanzadas de la sintaxis de la expresión regular pueden variar ligeramente. Consulte la documentación regex de cada lenguaje para más información.

Terminología

Vamos a usar la función **tokenize-regexp** para analizar la sintaxis de las expresiones regulares. Esta función divide texto en una secuencia de cadenas con ayuda de las expresiones regulares. Para ello la función toma estos parámetros de entrada:

input	La cadena de entrada que va a procesar la función. La expresión regular que operará en esta cadena.
pattern	La expresión regular pattern que se aplica.
flags	Este parámetro es opcional y define las demás opciones (flags) que determinan cómo se interpreta una expresión regular; consulte "Flags" más abajo.

En la asignación siguiente la cadena de entrada es "Altova MapForce". El parámetro **pattern** es un carácter de espacio y no se usan expresiones regulares flag.



Lo que ocurre es que se separa el texto cada vez que aparece el carácter espacio, por lo que el resultado de la asignación es:

```
<items>
   <item>Altova</item>
   <item>MapForce</item>
</items>
```

Observe que la función **tokenize-regexp** excluye los caracteres que coinciden del resultado. En otras palabras, el carácter espacio de este ejemplo se omite en el resultado.

El ejemplo anterior es muy básico, por lo que se puede conseguir el mismo resultado sin expresiones regulares, usando la función tokenize 316. En un caso real el parámetro **pattern** contendría una expresión regular más compleja. La expresión regular puede consistir en:

- Literales
- Clases de caracteres
- Intervalos de caracteres
- Clases negadas
- Metacaracteres
- Cuantificadores

Literales

Use literales para que los caracteres que coincidan sean idénticos a los que indica. Por ejemplo, si la cadena de entrada es **abracadabra** y **pattern** es el literal **br**, el resultado es:

```
<items>
   <item>a</item>
   <item>acada</item>
   <item>a</item>
</items>
```

La explicación es que el literal **br** encontró dos coincidencias en la cadena de entrada **abracadabra**. Una vez omitidos esos caracteres del resultado, queda la secuencia de tres cadenas que se ve en el extracto de código.

Clases de caracteres

Si encierra un conjunto de caracteres entre corchetes (**[** y **]**) se crea una clase de caracteres. Solamente coincide uno de los caracteres dentro de esa clase, por ejemplo:

- El patrón **[aeiou]** busca cualquier vocal en minúsculas.
- El patrón **[mj]ust** busca "must" y "just".

Nota: el patrón distingue entre mayúsculas y minúsculas. Por ejemplo, "a" no encontrará "A". Para que el patrón deje de distinguir entre mayúsculas y minúsculas debe usar el parámetro flag **i**, véase más abajo.

Intervalos de caracteres

Use **[a-z]** para crear el intervalo comprendido entre dos caracteres. Solamente se encontrará uno de los caracteres en cada búsqueda. Por ejemplo, el patrón **[a-z]** busca cualquier carácter de la a a la z que esté en minúsculas.

Clases negadas

Si usa el acento circunflejo (**^**) como primer carácter tras el corchete de apertura, se niega la clase de caracteres. Por ejemplo, el patrón **[^a-z]** busca cualquier carácter que no esté en esa clases de caracteres, incluidas las líneas nuevas.

Buscar cualquier carácter.

Use el metacarácter punto (**.**) para buscar un solo carácter, sea cual sea (excepto líneas nuevas). Por ejemplo, **.** busca un solo carácter, sea cual sea.

Cuantificadores

Dentro de una expresión regular, los cuantificadores definen cuántas veces debe aparecer el carácter o la subexpresión anteriores para que la búsqueda obtenga resultados.

?	Busca cero o una coincidencia de la cadena precedente (la cadena es opcional) Por ejemplo, el patrón `mo?` Encuentra "m" y "mo".
+	Busca una o más coincidencias de la cadena precedente. Por ejemplo, el patrón `mo+` Encuentra "mo", "moo", "mooo", etc.
*****	Busca cero o más coincidencias de la cadena precendente.
{min,max}	Busca el número de repeticiones entre *min* y *max*.. P. ej. `mo{1,3}`encuentra "mo", "moo", and "mooo".

Paréntesis

Los paréntesis `(` y `)` se usan para agrupar partes de la expresión regular. Se pueden usar para aplicar cuantificadores a una subexpresión (en vez de a un solo carácter) o con alternancia (véase a continuación).

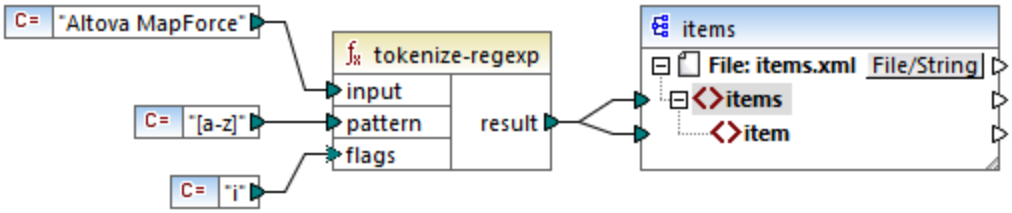
Alternancia

La barra vertical `|` se usa con el significado "o". Se puede usar para buscar cualquiera de las distintas subexpresiones separadas por `|`. Por ejemplo, el patrón `(horse|make) sense` busca tanto "horse sense" como "make sense".

Flags

Estos parámetros opcionales definen cómo se debe interpretar la expresión regular. Para establecer las opciones se usan letras, que pueden estar en cualquier orden y se pueden repetir.

s	Si está presente, el proceso de búsqueda opera en el modo "dot-all". Si la cadena de entrada input contiene "hello" y "world" en dos líneas *diferentes*, la expresión `hello*world` solamente encontrará resultados si se establece la marca flag **s**.
m	Si está presente, el proceso de búsqueda opera en el modo multilínea. En el modo multilínea el acento circunflejo `^` busca el principio de una línea, sea cual sea. Es decir, el inicio de una cadena entera y el primer carácter que aparece después del carácter de línea nueva. El carácter de dólar `$` busca el fin de una línea, sea cual sea. Es decir, el final de una cadena entera y el primer carácter que aparece antes del carácter de línea nueva. El carácter de línea nueva es **#x0A**..
i	Si está presente, el proceso de búsqueda no distingue entre mayúsculas y minúsculas. Por ejemplo, la expresión regular `[a-z]` más la marca **i** encontrará todas las letras de la a-z y entre

	A-Z. 
x	Si está presente, los caracteres de espacio en blanco se quitan de la expresión regular antes de iniciar el proceso de búsqueda. Los caracteres de espacio en blanco son **#x09**, **#x0A**, **#x0D** y **#x20**. **Nota:** los caracteres de espacio en blanco sin expresiones de clases de caracteres no se eliminan (por ejemplo, `[#x20]`).

6.6 Referencia de la biblioteca de funciones

En esta sección se describen todas las funciones integradas de MapForce que aparecen en la ventana Bibliotecas [21]. Estas funciones se organizan por bibliotecas. Las funciones disponibles en la ventana **Bibliotecas** dependen del lenguaje de transformación que haya escogido para la asignación. Para más información sobre los lenguajes de transformación disponibles consulte este apartado [17].

A continuación puede encontrar información sobre la compatibilidad de los lenguajes de transformación.

core | aggregate functions (agregado)

La lista siguiente es un resumen de la compatibilidad de las funciones principales con los lenguajes de transformación.

core | aggregate functions (agregado)

- **avg**, **max**, **max-string**, **min**, **min-string**: XSLT 2.0, XSLT 3.0, XQuery 1.0, C#, C++, Java, Built-In;
- **count**, **sum**: todos los lenguajes de transformación.

core | conversion functions (conversión)

- **boolean**, `string`, **number**: todos los lenguajes de transformación;
- **format-date**, **format-dateTime**, **format-time**: XSLT 2.0, XSLT 3.0, C#, C++, Java, Built-In;
- **format-number**: XSLT 1.0, XSLT 2.0, XSLT 3.0, C#, C++, Java, Built-In;
- **parse-date**, **parse-dateTime**, **parse-number**, **parse-time**: C#, C++, Java, Built-In.

core | file path functions (ruta de archivos)

Todas las funciones de ruta de archivos son compatibles con todos los lenguajes de transformación.

core | generator functions (generador)

La función **auto-number** está disponible para todos los lenguajes de transformación.

core | logical functions (lógica)

Las funciones de lógica son compatibles con todos los lenguajes de transformación.

core | math functions (matemáticas)

- **add**, **ceiling**, **divide**, **floor**, **modulus**, **multiply**, **round**, **subtract**: todos los lenguajes de transformación;
- **round-precision**: C#, C++, Java, Built-In.

core | node functions (nodo)

- **is-xsi-nil**, **local-name**, **static-node-annotation**, **static-node-name**: todos los lenguajes de transformación;
- **node-name**, **set-xsi-nil**, **substitute-missing-with-xsi-nil**: XSLT 2.0, XSLT 3.0, XQuery 1.0, C#, C++, Java, Built-In.

core | QName functions (QName)

Las funciones `Qname` son compatibles con todos los lenguajes de transformación excepto XSLT1.0.

core | sequence functions (secuencia)

- **exists**, **not-exists**, **position**, **substitute-missing**: todos los lenguajes de transformación;
- **distinct-values**, **first-items**, **generate-sequence**, **item-at**, **items-from-till**, **last-items**, **replicate-item**, **replicate-sequence**, **set-empty**, **skip-first-items**: XSLT 2.0, XSLT 3.0, XQuery 1.0, C#, C++, Java, Built-In;
- **group-adjacent**, **group-by**, **group-ending-with**, **group-into-blocks**, **group-starting-with**: XSLT 2.0, XSLT 3.0, C#, C++, Java, Built-In.

core | string functions (cadena)

- **concat**, **contains**, **normalize-space**, **starts-with**, **string-length**, **substring**, **substring-after**, **substring-before**, **translate**: todos los lenguajes de transformación;
- **char-from-code**, **code-from-char**, **tokenize**, **tokenize-by-length**, **tokenize-regexp**: XSLT 2.0, XSLT 3.0, XQuery 1.0, C#, C++, Java, Built-In.

Funciones bson (sólo en MapForce Enterprise)

Las funciones BSON son solamente compatibles con Built-In.

Funciones db (ediciones MapForce Professional y Enterprise)

Las funciones `db` son compatibles con C#, C++, Java y Built-In.

Funciones edifact (sólo en MapForce Enterprise Edition)

Las funciones `edifact` son compatibles con C#, C++, Java y Built-In.

Funciones lang (ediciones MapForce Professional y Enterprise)

La lista siguiente es un resumen de la compatibilidad de las funciones `lang` con los lenguajes de transformación.

lang | datetime functions (fechaHora)
Las funciones `lang | datetime` son compatibles con C#, C++, Java y Built-In.

lang | datetime functions (fechaHora)
Las funciones**read-binary-file** y **write-binary-file** sólo son compatibles con Built-In.

lang | generator functions (generador)
Las funciones **create-guid** son compatibles con C#, C++, Java y Built-In.

lang | logical functions (lógica)
Las funciones `lang | logical` son compatibles con C#, C++, Java y Built-In.

lang | math functions (matemáticas)
Las funciones `lang | math` son compatibles con C#, C++, Java y Built-In.

lang | QName functions (QName)
Las funciones `lang | QName` son compatibles con C#, C++, Java y Built-In.

como función de MapForce (en lang | QName functions)

- **charset-decode**, **charset-encode**: BUILT-IN
- **match-pattern**: C#, Java, Built-In.
- **capitalize**, **count-substring**, **empty**, **find-substring**, **format-guid-string**, **left**, **left-trim**, **lowercase**, **pad-string-left**, **pad-string-right**, **repeat-string**, **replace**, **reversefind-substring**, **right**, **right-trim**, **string-compare**, **string-compare-ignore-case**, **uppercase**: C#, C++, Java, Built-In.

Funciones functions (sólo en MapForce Enterprise)

Las funciones**mime** sólo son compatibles con Built-In.

Funciones xbrl (sólo en MapForce Enterprise)

Las funciones `xbrl` son compatibles con C#, C++, Java y Built-In.

Funciones xlsx (sólo en MapForce Enterprise)

Las funciones `xlsx` son compatibles con XSLT 2.0, XSLT 3.0, C#, Java y Built-In.

Funciones xpath2

Todas las funciones `xpath2` son compatibles con XSLT 2.0, XSLT 3.0 y XQuery 1.0.

Funciones xpath3

Las funciones `xpath3` son solamente compatibles con XSLT 3.0.

Funciones xslt10

La lista siguiente es un resumen de la compatibilidad de las funciones `xslt10` con los lenguajes de transformación.
xslt10 | xpath functions

- **local-name**, **name**, **namespace-uri**: XSLT 1.0, XSLT 2.0 y XSLT 3.0
- **lang**, **last**, **position**: XSLT 1.0

xslt10 | xslt functions

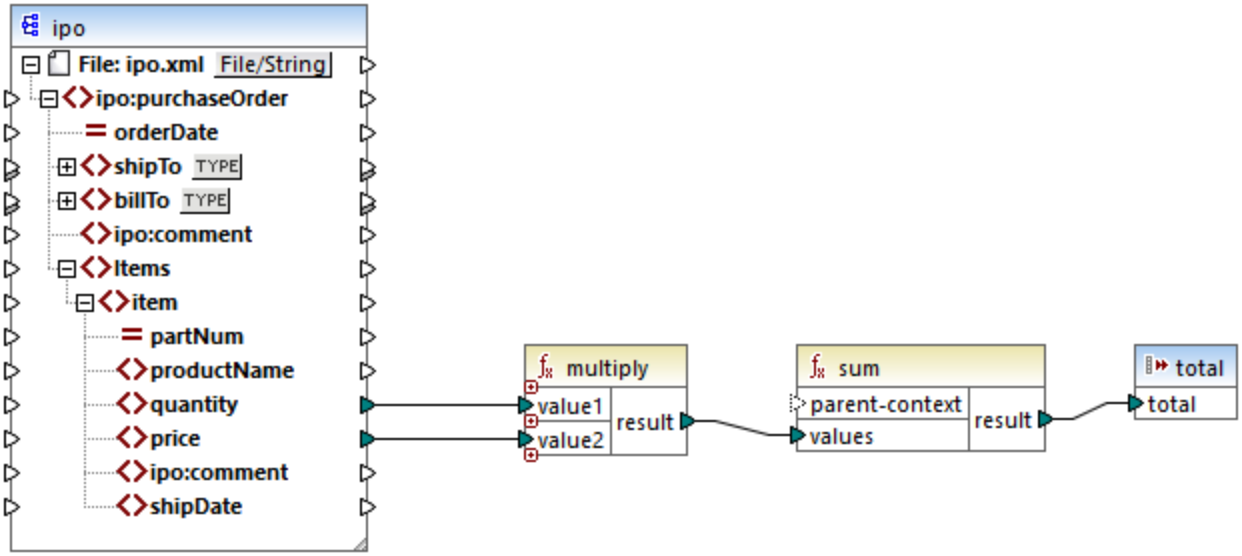
- **generate-id**, **system-property**: XSLT 1.0, XSLT 2.0 y XSLT 3.0
- **current**, **document**, **element-available**, **function-available**, **unparsed-entity-uri**: XSLT 1.0

6.6.1 core | aggregate functions (agregado)

Las funciones de agregado procesan varios valores del mismo tipo para obtener un resultado único, como una suma, un recuento o un promedio. Puede agregar datos en MapForce con ayuda de las funciones de agregado, como `avg`, `count`, `max`, etc.

Estos dos argumentos son comunes a todas las funciones de agregado.

1. **parent-context**. Este argumento es opcional y permite sobrescribir el contexto de asignación predeterminado (y con ello cambiar el alcance de la función o los valores que esta debe recorrer). Para ver un ejemplo, consulte Ejemplo: Cambiar el contexto primario 415.
2. **values**. Este argumento debe estar conectado a un elemento de entrada que suministra los datos que se van a procesar. Por ejemplo, en la asignación siguiente la función `sum` toma como entrada una secuencia de valores numéricos que proviene de un archivo XML de origen. La función `multiply` obtiene el precio multiplicado por la cantidad por cada elemento que hay en el archivo XML de origen y pasa el resultado a la función `sum`. La función `sum` agrega todos los valores de entrada y produce un resultado total que también es el resultado de la asignación. Puede encontrar esta asignación en el directorio **...\MapForceExamples\Tutorial**.

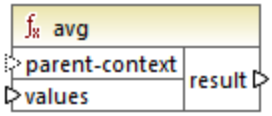


SimpleTotal.mfd

Algunas funciones de agregado, como `min`, `max`, `sum` o `avg`, sólo funcionan con valores numéricos. Los datos de entrada de estas funciones se convierten en tipo **decimal** para el procesamiento.

6.6.1.1 avg

Devuelve el valor promedio de todos los valores de la secuencia de entrada. El promedio de un conjunto de valores vacío es un conjunto vacío.



Lenguajes

Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

Parámetros

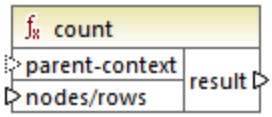
Argumento	Descripción
parent-context	Argumento opcional. Suministra el parent-context. Véase también el apartado Ejemplo: Cambiar el contexto primario 415.
values	Este argumento debe estar conectado a un elemento de entrada que suministra los datos propiamente dichos. Recuerde que el valor de argumento dado debe ser numérico.

Ejemplo

Consulte el apartado Ejemplo: agrupar registros por clave 287.

6.6.1.2 count

Devuelve el número de elementos que componen la secuencia de entrada. El recuento de un conjunto vacío es 0.



Lenguajes

Built-in, C++, C#, Java, XQuery#, XSLT 1.0 y, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Esta función no es totalmente compatible con XSLT1.

Parámetros

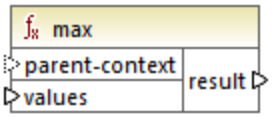
Argumento	Descripción
parent-context	Argumento opcional. Suministra el parent-context. Véase también el apartado Ejemplo: Cambiar el contexto primario 415.
nodos/filas	Este argumento debe estar conectado al elemento de origen que se debe contar.

Ejemplo

Consulte el apartado Ejemplo: Cambiar el contexto primario 415.

6.6.1.3 max

Devuelve el valor máximo de todos los valores numéricos de la secuencia de entrada. El valor máximo de un conjunto de valores vacío es un conjunto vacío.



Lenguajes

Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

Parámetros

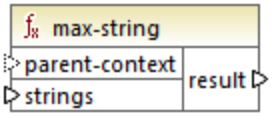
Argumento	Descripción
parent-context	Argumento opcional. Suministra el parent-context. Véase también el apartado Ejemplo: Cambiar el contexto primario 415.
values	Este argumento debe estar conectado a un elemento de entrada que suministra los datos propiamente dichos. Recuerde que el valor de argumento dado debe ser numérico. Para obtener el valor máximo de una secuencia de cadenas, use la función max-string 240 .

Ejemplo

Consulte el apartado Ejemplo: agrupar registros por clave 287.

6.6.1.4 max-string

Devuelve el valor máximo de todos los valores de cadena de la secuencia de entrada. Por ejemplo: `max-string("a", "b", "c")` devuelve `"c"`. Cuando se aplica a un conjunto vacío el resultado es una cadena vacía.



Lenguajes

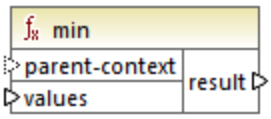
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

Parámetros

Argumento	Descripción
parent-context	Argumento opcional. Suministra el parent-context. Véase también el apartado Ejemplo: Cambiar el contexto primario 415.
strings	Este argumento debe estar conectado a un elemento de entrada que suministra los datos propiamente dichos. El valor de argumento dado debe ser una secuencia (cero o varios) de `xs:string`.

6.6.1.5 min

Devuelve el valor mínimo de todos los valores numéricos de la secuencia de entrada. El promedio de un conjunto de valores vacío es un conjunto vacío.



Lenguajes

Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

Parámetros

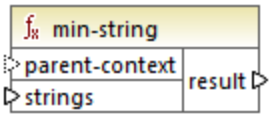
Argumento	Descripción
parent-context	Argumento opcional. Suministra el parent-context. Véase también el apartado Ejemplo: Cambiar el contexto primario 415.
values	Este argumento debe estar conectado a un elemento de entrada que suministra los datos propiamente dichos. Recuerde que el valor de argumento dado debe ser numérico. Para obtener el valor mínimo de un conjunto de cadenas, use la función min-string 242 .

Ejemplo

Consulte el apartado Ejemplo: agrupar registros por clave 287.

6.6.1.6 min-string

Devuelve el valor mínimo de todos los valores de cadena de la secuencia de entrada. Por ejemplo: **`min-string("a", "b", "c")`** devuelve **`"a"`**. Cuando se aplica a un conjunto vacío el resultado es una cadena vacía.



Lenguajes

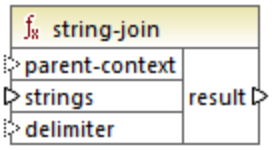
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

Parámetros

Argumento	Descripción
parent-context	Argumento opcional. Suministra el parent-context. Véase también el apartado Ejemplo: Cambiar el contexto primario 415.
strings	Este argumento debe estar conectado a un elemento de entrada que suministra los datos propiamente dichos. El valor de argumento dado debe ser una secuencia (cero o varios) de `xs:string`.

6.6.1.7 string-join

Enlaza todos los valores de la secuencia de entrada y forma una cadena delimitada por la cadena suministrada como delimitador al parámetro delimiter Cuando se aplica a un conjunto vacío el resultado es una cadena vacía.



Lenguajes

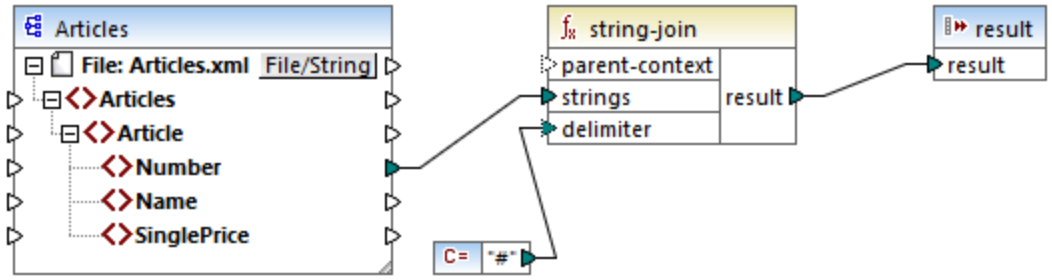
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

Parámetros

Argumento	Descripción
parent-context	Argumento opcional. Suministra el parent-context. Véase también el apartado Ejemplo: Cambiar el contexto primario 415.
strings	Este argumento debe estar conectado a un elemento de entrada que suministra los datos propiamente dichos. El valor de argumento dado debe ser una secuencia (cero o varios) de `xs:string`.
delimiter	Argumento opcional. Indica el delimitador que se introduce entre dos cadenas consecutivas.

Ejemplo

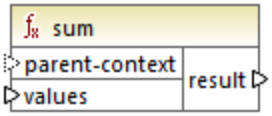
El ejemplo de la siguiente imagen muestra cuatro elementos **Article** con los números: 1, 2, 3, y 4.



La constante indica el carácter "#" como delimitador. El resultado de la asignación es, por tanto, **1#2#3#4**. Si no indica ningún delimitador el resultado sería **1234**.

6.6.1.8 sum

Devuelve la suma aritmética de todos los valores de la secuencia de entrada. La suma de un conjunto vacío es 0.



Lenguajes

Built-in, C++, C#, Java, XQuery#, XSLT 1.0 y, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Parámetros

Argumento	Descripción
parent-context	Argumento opcional. Suministra el parent-context. Véase también el apartado Ejemplo: Cambiar el contexto primario 415.
values	Este argumento debe estar conectado a un elemento de entrada que suministra los datos propiamente dichos. Recuerde que el valor de argumento dado debe ser numérico.

Ejemplo

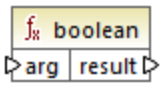
Consulte el apartado Ejemplo: sumar valores de nodo 227.

6.6.2 core | conversion functions (conversión)

MapForce ofrece varias funciones de conversión de tipos en la biblioteca **conversion** para realizar conversiones explícitas de tipos de datos. Tenga en cuenta que, en la mayoría de los casos, MapForce crea conversiones automáticamente y que solamente necesitará usar estas funciones en casos muy concretos. Si los nodos de entrada son de tipos diferentes (p. ej. entero y cadena), puede usar las funciones de conversión para forzar una comparación de cadenas o numérica. Por ejemplo, si en una asignación hay elementos de tipos distintos (como entero y cadena) puede usar la función de conversión number 253 para forzar una comparación numérica.

6.6.2.1 boolean

Convierte un valor numérico de entrada **arg** en un valor booleano. Esta función puede ser útil al trabajar con funciones lógicas (como `equal`, `greater`, etc.) además de con filtros y condiciones if-else 178. Para obtener un valor booleano**false** debe dar una cadena vacía o un valor numérico 0 como argumento. Para obtener un valor booleano **true** debe dar una cadena con contenido o un valor numérico 1 como argumento.



Lenguajes

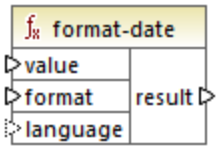
Built-in, C++, C#, Java, XQuery#, XSLT 1.0 y, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Parámetros

Argumento	Descripción
arg	Argumento obligatorio. Indica el valor que se va a convertir.

6.6.2.2 format-date

Convierte un valor de entrada `xs:date` en un valor de cadena y le aplica el formato indicado por las opciones especificadas.



Lenguajes

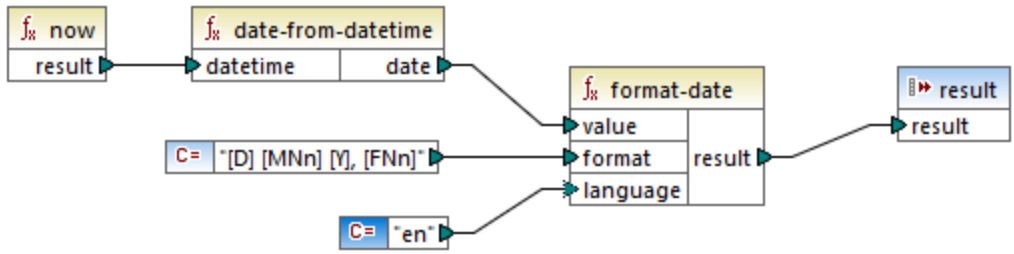
La función está disponible para XSLT 2.0, XSLT 3.0, Java, C#, C++ y el motor de ejecución integrado.

Parámetros

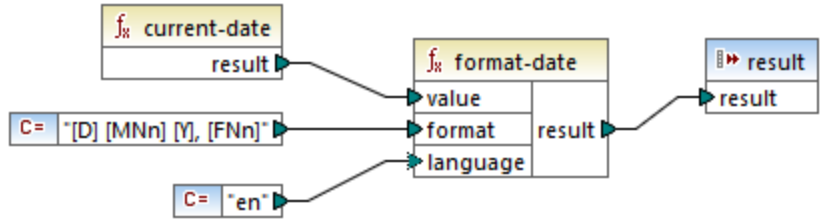
Argumento	Descripción
value	Fecha a la que se debe aplicar formato.
format	Cadena de formato que identifica el formato que se debe aplicar a la fecha. Este argumento se usa igual que el argumento **format** de la format-dateTime 246 .
language	Argumento opcional. Devuelve el nombre del mes y el día de la semana en el idioma seleccionado. Son valores válidos: **de** alemán **en (predet.)** inglés **es** español **fr** francés **ja** japonés

Ejemplo

En la asignación siguiente el resultado es la fecha actual en formato: "25 March 2020, Wednesday". Para traducir este valor al idioma español, defina el valor **es** para el argumento **language**.

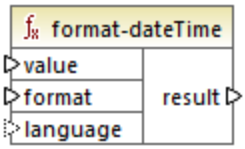


Tenga en cuenta que la asignación anterior se ha diseñado para Java, C#, C++ y el motor de ejecución integrado. Para obtener el mismo resultado en XSLT 2.0 use esta asignación:



6.6.2.3 format-dateTime

Convierte un valor de fecha y hora (`xs:dateTime`) en una cadena de texto. La representación de cadena de la fecha y hora sigue el formato que dicte el valor del argumento **format**



Lenguajes

La función está disponible para XSLT 2.0, XSLT 3.0, Java, C#, C++ y el motor de ejecución integrado.

Parámetros

Argumento	Descripción
value	El valor `xs:dateTime` al que se debe aplicar formato.
format	Cadena de formato que identifica el formato que se debe aplicar al **valor**. Véanse las "Observaciones", más abajo
language	Argumento opcional. Devuelve el nombre del mes y el día de la semana en el idioma seleccionado. Son valores válidos:

Argumento	Descripción
	de alemán **en (predet.)** inglés **es** español **fr** francés **ja** japonés

Nota: si el resultado de la función (result) se conecta a un nodo de un tipo que no sea string, el formato se puede perder porque el valor se convierte al tipo de destino. Esta conversión automática se puede deshabilitar desactivando la casilla **Convertir valores en tipos de destino** en el cuadro de diálogo "Configuración" del componente de destino (véase Cambiar la configuración 39 de los componentes.

Observaciones

El argumento **format** está compuesto por una cadena que lleva los llamados marcadores de variable entre corchetes. Los caracteres situados fuera de los corchetes son caracteres literales que se deben copiar en el resultado. Si necesita usar corchetes como caracteres literales en el resultado, escríbalos dos veces.

Cada marcador de variable está compuesto por (i) un especificador de componente que identifica qué componente de fecha u hora se debe mostrar, (ii) un modificador de formato opcional, (iii)otro modificador de presentación opcional y (iv) un modificador de ancho opcional precedido de una coma, si existe.

```
format := (literal | argument)*
argument := [component(format)?(presentation)?(width)?]
width := , min-width ("-" max-width)?
```

Estos son los componentes:

Especificador	Descripción	Presentación predeterminada
Y	año (valor absoluto)	cuatro dígitos (2010)
M	mes del año	1-12
D	día del mes	1-31
d	día del año	1-366
F	día de la semana	nombre del día (dependiendo del idioma)
W	semana del año	1-53
w	semana del mes	1-5
H	hora (24 horas)	0-23

Especificador	Descripción	Presentación predeterminada
h	hora (12 horas)	1-12
P	A.M. o P.M.	alfabética (dependiendo del idioma)
m	minutos de una hora	00-59
s	segundos de un minuto	00-59
f	segundos fraccionarios	numérica, con un decimal
Z	uso horario como diferencia horaria de UTC	+08:00
z	uso horario como diferencia horaria usando GMT	GMT+n

El modificador de formato puede ser uno de estos:

Carácter	Descripción	Ejemplo
1	formato decimal numérico sin ceros iniciales: 1, 2, 3, ...	1, 2, 3
01	formato decimal, con dos dígitos: 01, 02, 03, ...	01, 02, 03
N	nombre del componente, todo en mayúsculas	LUNES, MARTES 1)
n	nombre del componente, todo en minúsculas	lunes, martes 1)
Nn	nombre del componente, primera letra en mayúsculas	Lunes, Martes 1)

Notas:

1. los modificadores **N**, **n** y **Nn** solamente son compatibles con estos componentes: **M**, **d**, **D**.

El modificador del ancho, si es necesario, viene introducido por una coma y seguido de un dígito que exprese el ancho mínimo. Opcionalmente puede añadir un guion seguido de otro dígito para expresar el ancho máximo. Por ejemplo:

- **`[D,2]`** es el día del mes, con ceros iniciales (dos dígitos).
- **`[MNn,3-3]`** es el nombre del mes, escrito con tres caracteres, por ejemplo *Ene*, *Feb*, *Mar*, etc.

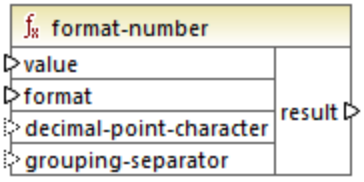
Ejemplos

La tabla siguiente muestra algunos ejemplos de formato de valores `xs:dateTime`, obtenido con ayuda de la función **`format-dateTime`**. La columna Valor de la tabla especifica el valor dado al argumento **value**. La columna Formato de la tabla especifica el valor del argumento **format**. La columna Resultado muestra el valor que devuelve la función.

Valor	Formato	Resultado
2003-11-03T00:00:00	`[D]/[M]/[Y]`	11/03/2003
2003-11-03T00:00:00	`[Y]-[M,2]-[D,2]`	03/11/2003
2003-11-03T00:00:00	`[Y]-[M,2]-[D,2] [H,2]:[m]:[s]`	2003-11-03 00:00:00
2010-06-02T08:02	`[Y] [MNn] [D01] [F,3-3] [d] [H]:[m]:[s].[f]`	2010 June 02 Wed 153 8:02:12.054
2010-06-02T08:02	`[Y] [MNn] [D01] [F,3-3] [d] [H]:[m]:[s].[f] [z]`	2010 June 02 Wed 153 8:02:12.054 GMT+02:00
2010-06-02T08:02	`[Y] [MNn] [D1] [F] [H]:[m]:[s].[f] [Z]`	2010 June 2 Wednesday 8:02:12.054 +02:00
2010-06-02T08:02	`[Y] [MNn] [D] [F,3-3] [H01]:[m]:[s]`	2010 June 2 Wed 08:02:12

6.6.2.4 format-number

Convierte un número en una cadena y le aplica el formato indicado .en las opciones.



Lenguajes

Built-in, C++, C#, Java#, XSLT 1.0 y, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Parámetros

Argumento	Descripción
value	Argumento obligatorio. Suministra el número al que se debe aplicar formato.
format	Argumento obligatorio. Suministra una cadena de formato que indica de qué forma se debe dar formato al número. Véanse las "Observaciones", más abajo
decimal-point-format	Argumento opcional. Suministra el carácter que se debe usar como carácter de punto decimal. El valor predeterminado es el carácter de punto

Argumento	Descripción
	(.).
grouping-separator	Argumento opcional. Suministra el carácter que se debe usar para separa grupos de números. El valor predeterminado es el carácter de coma (,).

Nota: si el resultado de la función (result) se conecta a un nodo de un tipo que no sea string, el formato se puede perder porque el valor se convierte al tipo de destino. Esta conversión automática se puede deshabilitar desactivando la casilla **Convertir valores en tipos de destino** en el cuadro de diálogo "Configuración" del componente de destino (véase Cambiar la configuración 39 de los componentes.

Observaciones

El argumento **format** toma esta forma:

```
format := subformat (;subformat)?
 subformat := (prefix)? integer (.fraction)? (suffix)?
 prefix := any characters except special characters
 suffix := any characters except special characters
 integer := (#)* (0)* ( allowing ',' to appear)
 fraction := (0)* (#)* (allowing ',' to appear)
```

El primer *subformat* se utiliza para dar formato a los números positivos y el segundo para los números negativos. Si solamente se da un *subformat*, se utiliza el mismo para los números negativos, pero con un signo - antes del *prefix*.

Carácter especial	Predeterminado	Descripción
dígito cero	0	en este punto del resultado siempre aparecerá un dígito
dígito	#	en este punto del resultado siempre aparecerá un dígito, excepto si es un cero inicial o final no significativo
punto decimal	.	separa el entero y la parte de fracción del número
grouping-separator	,	separa grupos de dígitos
signo de porcentaje	%	multiplica el número por 100 y lo muestra como porcentaje
por mil	‰	multiplica el número por 1000 y lo muestra como por mil

Los caracteres utilizados por el carácter de punto decimal y de separador de grupos siempre son "." y "," respectivamente.

Nota: el método de redondeo utilizado para esta función es al alza (p. ej. redondea al alza si la fracción es mayor o igual a 0.5 y redondea a la baja si la fracción es menor que 0.5). Este método de redondeo solamente afecta al código generado y si la opción seleccionada es el motor de ejecución integrado.

El XSLT 1.0 el modo de redondeo es indeterminado. En XSLT 2.0 el modo de redondeo es round-half-to-even, es decir se redondea al número par más próximo.

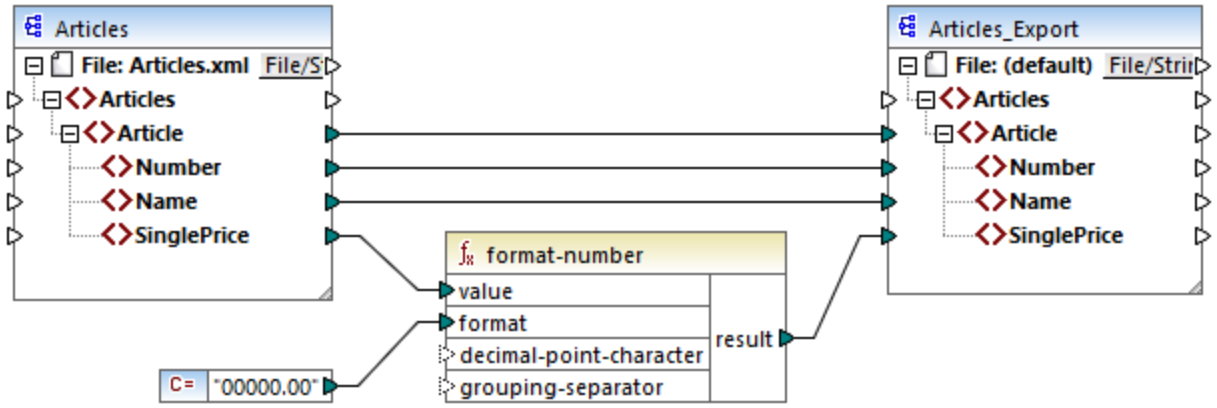
Número	Cadena de formato	Resultado
1234,5	#,##0.00	1.234,50
123,456	#,##0.00	123,46
1000000	#,##0.00	1.000.000,00
-59	#,##0.00	-59,00
1234	###0.0###	1234,0
1234,5	###0.0###	1234,5
.00025	###0.0###	0,0003
.00035	###0.0###	0,0004
0,25	#00%	25%
0,736	#00%	74%
1	#00%	100%
-42	#00%	-4200%
-3,12	#.00;(#.00)	(3.12)
-3,12	#.00;#.00CR	3.12CR

Ejemplo

La asignación siguiente lee datos de un archivo XML de origen y los escribe en un archivo XML de destino. Existen varios elementos **SinglePrice**en el archivo de origen que contienen estos valores decimales: **25**, **2.30**, **34**, **57.50**. La asignación tiene dos objetivos:

1. Añadir ceros a la izquierda de todos los valores de forma que la parte entera ocupe exactamente 5 dígitos
2. Añadir ceros a la derecha de todos los valores de forma que la parte decimal ocupe exactamente 2 dígitos

Para ello se da la cadena de formato `00000.00` como argumento a la función `format-number`.



PreserveFormatting.mfd

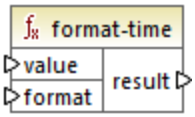
En consecuencia, los valores de destino paran a ser:

```
00025,00
00002,30
00034,00
00057,50
```

Puede encontrar el archivo de diseño de esta asignación en la ruta:
<Documentos>\Altova\MapForce2024\MapForceExamples\PreserveFormatting.mfd.

6.6.2.5 format-time

Convierte un valor `xs:time` de entrada en una cadena.



Lenguajes

La función está disponible para XSLT 2.0, XSLT 3.0, Java, C#, C++ y el motor de ejecución integrado.

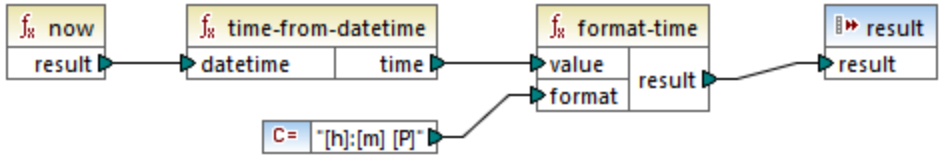
Parámetros

Argumento	Descripción
value	Argumento obligatorio. Suministra el valor `xs:time` para darle formato.
format	Argumento obligatorio. Indica el formato de la cadena. Este argumento se usa igual que el argumento **format** de la format-dateTime 246 .

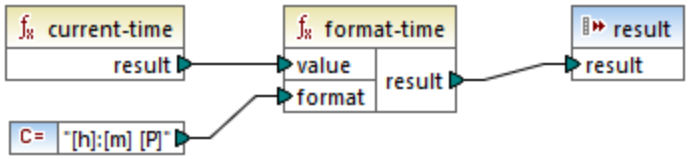
Ejemplo

En la asignación siguiente el resultado es la hora actual en formato `2:15 p.m.` . Para ello se usa la cadena de formato `[h]:[m] [P]`, donde:

- **[h]** es la hora actual en formato 12 horas
- **[m]** es el minuto de la hora actual
- **[P]** indica si es "a.m." o "p.m."

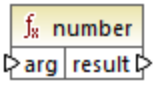


Tenga en cuenta que la asignación anterior se ha diseñado para Java, C#, C++ y el motor de ejecución integrado. Para obtener el mismo resultado en XSLT 2.0 use esta asignación:



6.6.2.6 number

Convierte el valor de **arg** en un número, donde **arg** es una cadena de valor booleano. Si **arg** es una cadena, MapForce intentará analizarlo como número. Por ejemplo, una cadena como `"12.56"` se convierte en el valor decimal `12.56`. Si **arg** tiene el valor booleano **true**, se convierte en el valor numérico **1**. Si **arg** tiene el valor booleano **false** , se convierte en el valor numérico **0**.



Lenguajes

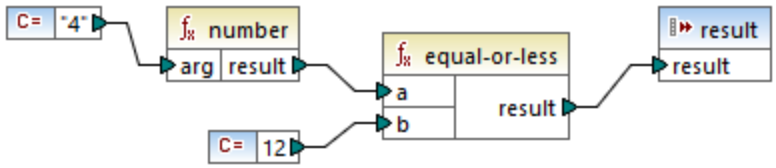
Built-in, C++, C#, Java, XQuery#, XSLT 1.0 y, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Parámetros

Argumento	Descripción
arg	Argumento obligatorio. Indica el valor que se va a convertir.

Ejemplo

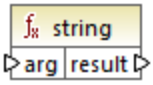
En el ejemplo siguiente la primera constante es de tipo `string` y contiene la cadena "4". La segunda constante contiene la constante numérica 12. Para que los dos valores se puedan comparar como números deben ser del mismo tipo.



Si añade una función **number** a la primera constante, la cadena "4" se convierte en el valor numérico 4. Es decir, que el resultado de la comparación es "true". Si no se usara la función **number** (es decir, si "4" se conectara directamente a **a**), se llevaría a cabo una comparación de cadenas y el resultado sería "false".

6.6.2.7 string

Convierte un valor de entrada en una cadena. La función también se puede usar para recuperar el contenido de texto de un nodo. Si el nodo de entrada es un tipo complejo XML, todos los descendientes también se generan como una sola cadena.



Lenguajes

Built-in, C++, C#, Java, XQuery#, XSLT 1.0 y, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Parámetros

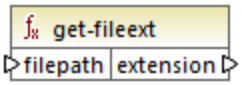
Argumento	Descripción
arg	Argumento obligatorio. Indica el valor que se va a convertir.

6.6.3 core | file path functions (ruta de archivos)

Las funciones de la biblioteca **file path** sirven para acceder a datos de una ruta de acceso (como carpetas, nombres de archivos y extensiones) y manipularlos para después procesarlos en la asignación. Estas funciones están disponibles para todos los lenguajes compatibles con MapForce.

6.6.3.1 get-fileext

Devuelve la extensión de la ruta de archivo, incluido el carácter "."



Lenguajes

Built-in, C++, C#, Java, XQuery#, XSLT 1.0 y, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Parámetros

Argumento	Descripción
filepath	Argumento obligatorio. Suministra la ruta de acceso al archivo que se quiere procesar.

Ejemplo

P. ej. c:\data\Sample.mfd devuelve ".mfd"

6.6.3.2 get-folder

Devuelve el nombre de la carpeta de la ruta de archivo, incluida la barra diagonal final o la barra diagonal inversa final.

Lenguajes

Built-in, C++, C#, Java, XQuery#, XSLT 1.0 y, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Parámetros

Argumento	Descripción
filepath	Argumento obligatorio. Suministra la ruta de acceso al archivo que se quiere procesar.

Ejemplo

P. ej. c:/data/Sample.mfd devuelve c:/data/Sample.

6.6.3.3 main-mfd-filepath

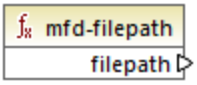
Devuelve la ruta completa del archivo mfd que contiene la asignación principal. Si el archivo mfd está sin guardar, la función devuelve una cadena vacía.

Lenguajes

Built-in, C++, C#, Java, XQuery#, XSLT 1.0 y, XSLT 1.0, XSLT 2.0, XSLT 3.0.

6.6.3.4 mfd-filepath

Si se le llama desde la asignación principal, la función devuelve el mismo resultado que la función main-mfd-filepath 256 , es decir, la ruta completa del archivo mfd que contiene la asignación principal. Si el archivo mfd archivo está sin guardar, la función devuelve una cadena vacía. Si se le llama desde dentro de una función definida por el usuario que es *importada* por un archivo mfd, la función mfd-filepath devuelve la ruta completa del archivo mfd *importado* que contiene la definición de la función definida por el usuario.

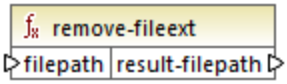


Lenguajes

Built-in, C++, C#, Java, XQuery#, XSLT 1.0 y, XSLT 1.0, XSLT 2.0, XSLT 3.0.

6.6.3.5 remove-fileext

Borra la extensión de la ruta de archivo, incluido el carácter "."



Lenguajes

Built-in, C++, C#, Java, XQuery#, XSLT 1.0 y, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Parámetros

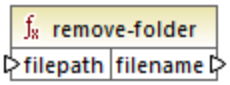
Argumento	Descripción
filepath	Argumento obligatorio. Suministra la ruta de acceso al archivo que se quiere procesar.

Ejemplo

P. ej. c:/data/Sample.mfd devuelve c:/data/Sample.

6.6.3.6 remove-folder

Borra el directorio de la ruta de acceso, incluida la barra diagonal final o la barra diagonal inversa final.



Lenguajes

Built-in, C++, C#, Java, XQuery#, XSLT 1.0 y, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Parámetros

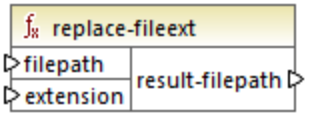
Argumento	Descripción
filepath	Argumento obligatorio. Suministra la ruta de acceso al archivo que se quiere procesar.

Ejemplo

P. ej. c:\data\Sample.mfd devuelve ".mfd"

6.6.3.7 replace-fileext

Reemplaza la extensión de la ruta de acceso dada por el parámetro **filepath** con la extensión dada por el parámetro **extension**.



Lenguajes

Built-in, C++, C#, Java, XQuery#, XSLT 1.0 y, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Parámetros

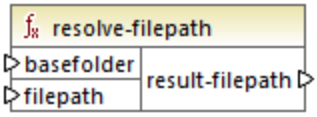
Argumento	Descripción
filepath	Argumento obligatorio. Suministra la ruta de acceso al archivo que se quiere procesar.
extension	Argumento obligatorio. Suministra la estensión nueva que se debe usar.

Ejemplo

P. ej. si el parámetro **filepath** es la ruta de acceso c:/data/Sample.mfd y .mfp es la extensión del parámetro **extension**, la función devuelve `c:\data\Sample.txt`.

6.6.3.8 resolve-filepath

Convierte una ruta de acceso relativa en una carpeta base relativa o absoluta. La función admite "." (directorio actual) y ".." (directorio primario).



Lenguajes

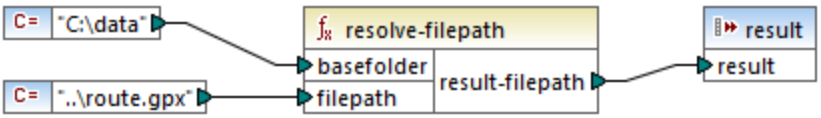
Built-in, C++, C#, Java, XQuery#, XSLT 1.0 y, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Parámetros

Argumento	Descripción
basefolder	Argumento obligatorio. Suministra el directorio base relativo en el que se debe resolver la ruta, que puede ser absoluta o relativa.
filepath	Argumento obligatorio. Suministra la ruta de acceso relativa que se quiere resolver.

Ejemplos

En la imagen siguiente, la ruta relativa **..\route.gpx** se resuelve con respecto al directorio **C:\data**.



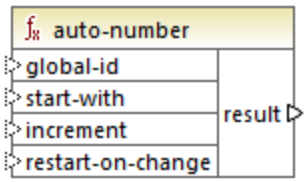
El resultado de la asignación es `C:\route.gpx`.

6.6.4 core | generator functions (generador)

La biblioteca de funciones **core / generator** incluye funciones que generan valores.

6.6.4.1 auto-number

La función auto-number genera enteros en una secuencia (por ejemplo, 1, 2, 3, 4...). Puede usar parámetros para configurar el entero de inicio, el valor incremental y otras opciones.



El orden exacto en el que la asignación llama a las funciones es indefinido. Puede que MapForce necesite guardar en la memoria caché los resultados calculados para volver a utilizarlos o evaluar las expresiones en cualquier orden. Al contrario que otras funciones, la función **auto-number** devuelve un resultado distinto si se la llama varias veces con los mismos parámetros de entrada. Por lo tanto, recomendamos que use esta función con cuidado. En algunos casos se puede conseguir el mismo resultado utilizando la función position 300 en lugar de auto-number.

Lenguajes

Built-in, C++, C#, Java, XQuery#, XSLT 1.0 y, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Parámetros

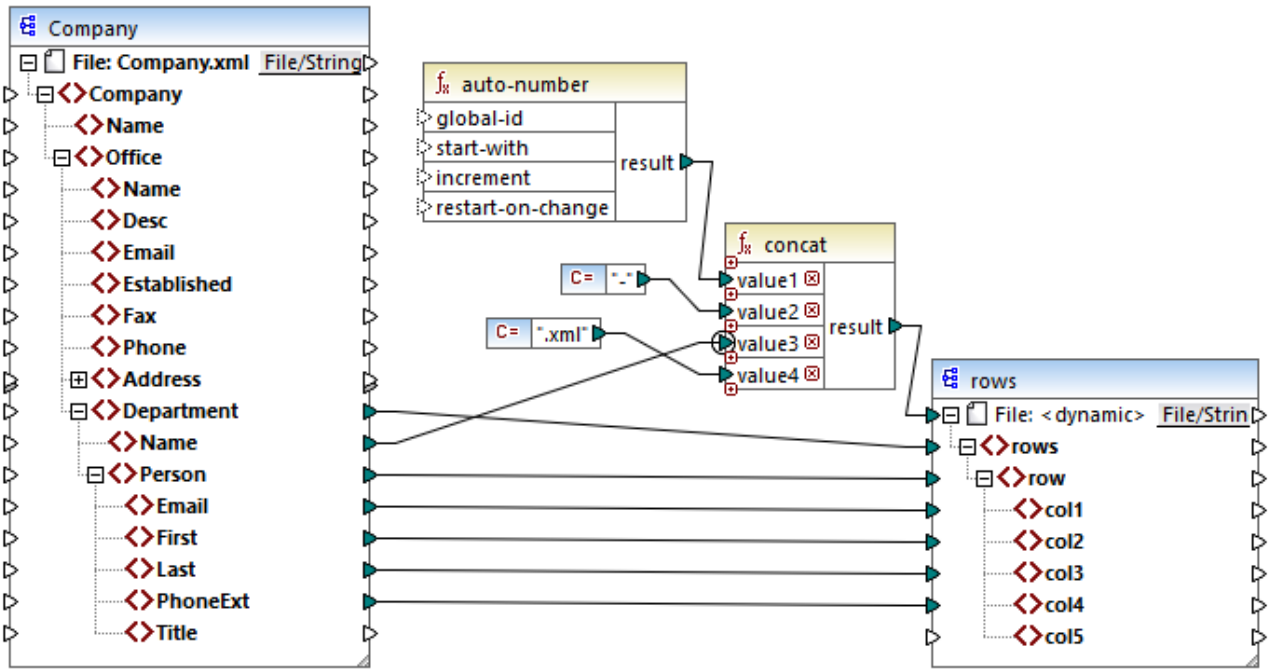
Argumento	Descripción
global-id	Parámetro opcional. Si un diseño de asignación contiene varias funciones **auto-number**, estas generarán secuencias con números duplicados (solapados). Para que todas las funciones **auto-number** sean conscientes unas de otras y no generen secuencias que se solapen, debe conectar una cadena común (por ejemplo, una constante) a la entrada **global-id** de cada función **auto-number**.

Argumento	Descripción
start-with	Parámetro opcional. Indica el entero con el que empieza la secuencia que se genera. El valor predeterminado es **1**.
increment	Parámetro opcional. Indica el valor incremental. El valor predeterminado es **1**.
restart-on-change	Parámetro opcional. Restablece el contador a **start-with** cuando cambia el contenido del elemento conectado.

Ejemplo

Esta asignación es una variación de la asignación **ParentContext.mfd** Ejemplo: Cambiar el contexto primario 415.

El objetivo de la asignación siguiente es generar varios archivos XML, uno por cada departamento del archivo XML de origen. Hay varios departamentos con el mismo nombre (esto se debe a que pertenecen a distintas oficinas centrales). Por este motivo, cada nombre de archivo que se genere debe empezar por un número secuencial, por ejemplo **1-Administration.xml**, **2-Marketing.xml**, etc.



Para conseguir el objetivo de la asignación se usó la función **auto-number**. El resultado de esta función se concatena con un guion seguido por el nombre del departamento, seguido por la cadena ".xml" para crear el nombre único del archivo generado. Es importante que se aplique un contexto de prioridad al tercer parámetro de la función **concat** nombre del 418 departamento). Esto sirve para llamar a la función **auto-number** en el contexto de cada departamento y produce los valores secuenciales necesarios. Sin ese contexto de prioridad, la función **auto-number** seguiría generando el número number 1 y, en consecuencia, se generarían también nombres de archivo duplicados.

6.6.5 core | logical functions (lógica)

Las funciones lógicas de la biblioteca core | logical se usan por lo general para comparar datos de entrada y su resultado son los valores booleanos true o false. Estas funciones se suelen utilizar para probar datos antes de pasarlos, por medio de un filtro, 178 a un subconjunto del componente de destino. Casi todas las funciones lógicas tienen esta estructura:

Parámetros de entrada = `a | b` o `value1 | value2`
Parámetro de salida = `result`

El resultado de la evaluación de dos nodos de entrada depende de los valores de entrada así como de los tipos de datos utilizados para la comparación. Por ejemplo, la comparación "less than" de los valores enteros **4** y **12** da como resultado el valor booleano "true" porque 4 es menor que 12. Si las dos cadenas de entrada contienen **"4"** y **"12"**, entonces el análisis léxico da como resultado el valor de salida "false" porque "4" es alfabéticamente mayor que el primer carácter "1" del segundo operando (12).

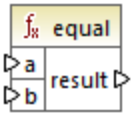
Si todos los tipos de datos de entrada son del mismo tipo (p. ej. todos los nodos de entrada son de tipo numérico o son cadenas), entonces la comparación se hace para el tipo común. Si los nodos de entrada son de tipos distintos (p. ej. un `integer` y una `cadena` o una `cadena` y una `fecha`), entonces el tipo de datos utilizado para la comparación es el tipo de datos de entrada más general y menos limitado de los dos.

Antes de iniciarse la comparación de los dos valores, todos los valores de entrada se convierten en el tipo de datos común. Por ejemplo, el tipo de datos "string" es menos limitado que "integer". La comparación del valor entero **4** con la cadena **"12"** convierte el valor entero **4** en la cadena **"4"**, que después se compara con la cadena **"12"**.

Nota: las funciones lógicas no se pueden usar para comprobar la existencia de valores null. Si se da un valor null como argumento de una función lógica, la función devuelve un valor null. Para más información consulte el apartado Valores Nil y Nillable) 121.

6.6.5.1 equal

El resultado es `true` si **a** es igual a **b**. De lo contrario devuelve `false`. La comparación distingue entre mayúsculas y minúsculas.



Ejemplo:

```
a = hi
b = hi
```

En este ejemplo los dos valores son iguales, por lo que el resultado es `true`. Si, por ejemplo, `b` fuera igual que `Hi`, la función devolvería `false`.

Lenguajes

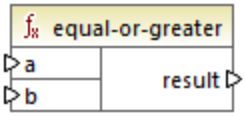
Built-in, C++, C#, Java, XQuery#, XSLT 1.0 y, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Parámetros

Argumento	Descripción
a	Parámetro obligatorio. Suministra el primer valor de la comparación.
b	Parámetro obligatorio. Suministra el segundo valor de la comparación.

6.6.5.2 equal-or-greater

El resultado es **true** si *a* es igual/mayor que *b*. De lo contrario, el resultado es **false**.



Lenguajes

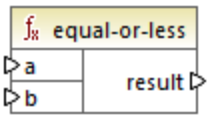
Built-in, C++, C#, Java, XQuery#, XSLT 1.0 y, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Parámetros

Argumento	Descripción
a	Parámetro obligatorio. Suministra el primer valor de la comparación.
b	Parámetro obligatorio. Suministra el segundo valor de la comparación.

6.6.5.3 equal-or-less

El resultado es **true** si *a* es igual/menor que *b*. De lo contrario, el resultado es **false**.



Lenguajes

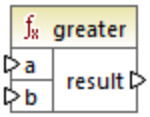
Built-in, C++, C#, Java, XQuery#, XSLT 1.0 y, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Parámetros

Argumento	Descripción
a	Parámetro obligatorio. Suministra el primer valor de la comparación.
b	Parámetro obligatorio. Suministra el segundo valor de la comparación.

6.6.5.4 greater

El resultado es **true** si *a* es mayor que *b*. De lo contrario devuelve **false**.



Lenguajes

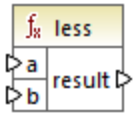
Built-in, C++, C#, Java, XQuery#, XSLT 1.0 y, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Parámetros

Argumento	Descripción
a	Parámetro obligatorio. Suministra el primer valor de la comparación.
b	Parámetro obligatorio. Suministra el segundo valor de la comparación.

6.6.5.5 less

El resultado es **true** si *a* es menor que *b*. De lo contrario devuelve **false**.



Lenguajes

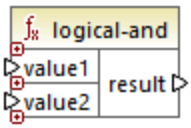
Built-in, C++, C#, Java, XQuery#, XSLT 1.0 y, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Parámetros

Argumento	Descripción
a	Parámetro obligatorio. Suministra el primer valor de la comparación.
b	Parámetro obligatorio. Suministra el segundo valor de la comparación.

6.6.5.6 logical-and

Si los dos valores de entrada son **true**, el resultado es **true**. Si son diferentes, el resultado es **false**. Puede conectar el resultado a otra función `logical-and` de modo que puede conectar varias condiciones con una conjunción lógica Y para comprobar si todas ellas devuelven **true**. Esta función se puede ampliar para que acepte más argumentos, consulte Agregar o eliminar argumentos en una función 198.



Lenguajes

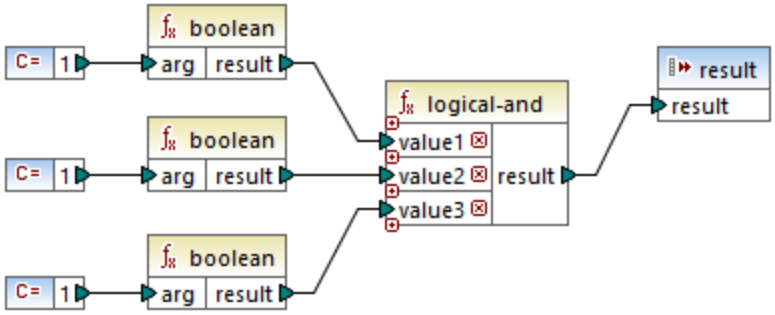
Built-in, C++, C#, Java, XQuery#, XSLT 1.0 y, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Parámetros

Argumento	Descripción
value1	Parámetro obligatorio. Suministra el primer valor de la comparación.
value2	Parámetro obligatorio. Suministra el segundo valor de la comparación.

Ejemplo

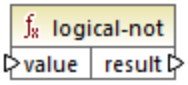
La asignación siguiente devuelve **true** porque todos los valores de entrada de la función `logical-and` también son **true**. Si cualquiera de los valores de entrada fuera **false**, entonces el resultado de la asignación también sería **false**.



Véase también el apartado Ejemplo: búsqueda y concatenación 219.

6.6.5.7 logical-not

Invierte o le da la vuelta al estado lógico/resultado. Si el *valor* de entrada es **true**, el resultado de la función es **false**. Si el valor de entrada es **true**, el resultado de la función es **false**.



Lenguajes

Built-in, C++, C#, Java, XQuery#, XSLT 1.0 y, XSLT 1.0, XSLT 2.0, XSLT 3.0.

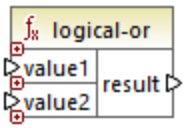
Parámetros

Argumento	Descripción
value	Parámetro obligatorio. Suministra el valor de entrada.

6.6.5.8 logical-or

Esta función exige que ambos valores de entrada sean booleanos. Si uno de los dos valores de entrada (ya sea value1 o value2) de la función logical-or es **true**, el resultado de la función es **true**. Si ambos son **false**, el resultado es **false**.

Esta función se puede ampliar para que acepte más argumentos, consulte Agregar o eliminar argumentos en una función 198.



Lenguajes

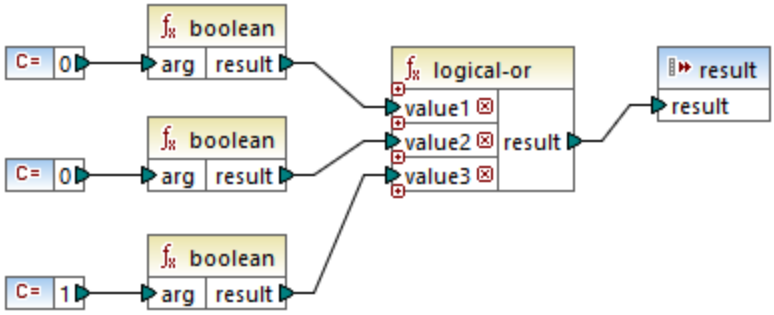
Built-in, C++, C#, Java, XQuery#, XSLT 1.0 y, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Parámetros

Argumento	Descripción
value1	Parámetro obligatorio. Suministra el primer valor de la comparación.
value2	Parámetro obligatorio. Suministra el segundo valor de la comparación.

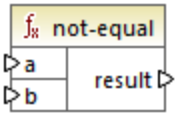
Ejemplo

El resultado de la asignación siguiente es **true** porque al menos uno de los argumentos de la función es **true**.



6.6.5.9 not-equal

El resultado es **true** si *a* no es igual a *b*. De lo contrario devuelve **false**.



Lenguajes

Built-in, C++, C#, Java, XQuery#, XSLT 1.0 y, XSLT 1.0, XSLT 2.0, XSLT 3.0.

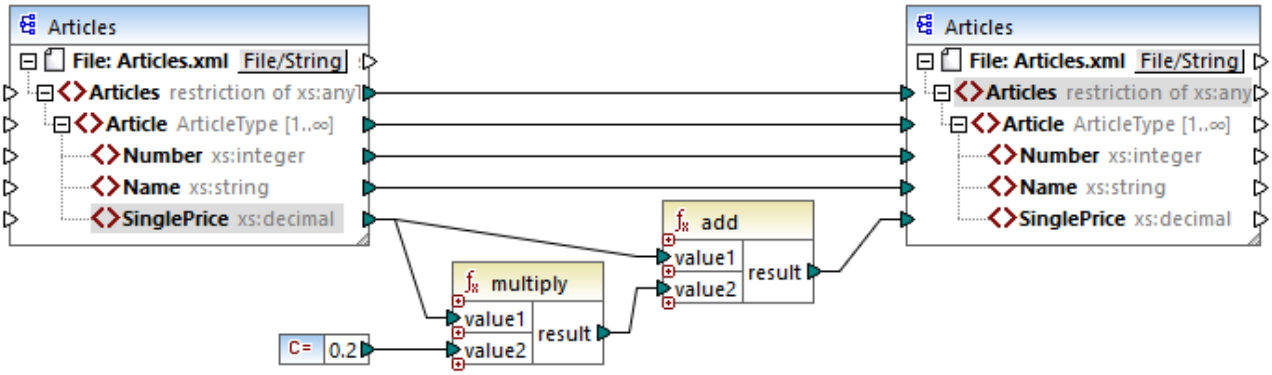
Parámetros

Argumento	Descripción
a	Parámetro obligatorio. Suministra el primer valor de la comparación.
b	Parámetro obligatorio. Suministra el segundo valor de la comparación.

6.6.6 core | math functions (matemáticas)

Las funciones matemáticas de la biblioteca core | math sirven para realizar operaciones matemáticas básicas con los datos. Tenga en cuenta que no se pueden usar para cálculos con datos `duration` ni `datetime`.

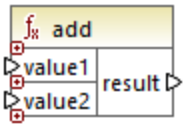
La mayoría de las funciones matemáticas toman dos parámetros de entrada (**value1** y **value2**) como operandos. Los valores de entrada se convierten automáticamente en tipo `decimal` para continuar el procesamiento. El resultado de las funciones matemáticas también es de tipo `decimal`



En la asignación de esta imagen, por ejemplo, se añade un 20% de impuestos sobre las ventas en los artículos asignados al componente de destino.

6.6.6.1 add

El resultado es el valor decimal resultante de sumar **value1** a **value2**. Esta función se puede ampliar para que acepte más argumentos, consulte Agregar o eliminar argumentos en una función 198.



Lenguajes

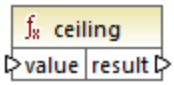
Built-in, C++, C#, Java, XQuery#, XSLT 1.0 y, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Parámetros

Argumento	Descripción
value1	Parámetro obligatorio. Suministra el primer operando
value2	Parámetro obligatorio. Suministra el segundo operando

6.6.6.2 ceiling

El resultado es el entero más pequeño que es mayor o igual que **value**.



Lenguajes

Built-in, C++, C#, Java, XQuery#, XSLT 1.0 y, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Parámetros

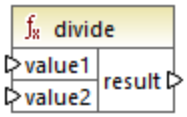
Argumento	Descripción
value	Parámetro obligatorio. Suministra el valor de entrada de la función.

Ejemplo

P. ej. si el resultado de una función de división es **11.2** y a este resultado le aplicamos la función `ceiling`, el resultado se convierte en **12** (es decir, el número entero mayor más cercano).

6.6.6.3 divide

El resultado es el valor decimal resultante de dividir **value1** por **value2**. La precisión del resultado depende del lenguaje de destino. Utilice la función round-precision 271 para definir la precisión del resultado.



Lenguajes

Built-in, C++, C#, Java, XQuery#, XSLT 1.0 y, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Parámetros

Argumento	Descripción
value1	Parámetro obligatorio. Suministra el primer operando
value2	Parámetro obligatorio. Suministra el segundo operando

6.6.6.4 floor

El resultado es el entero mayor que sea mayor o igual que **value**.

Lenguajes

Built-in, C++, C#, Java, XQuery#, XSLT 1.0 y, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Parámetros

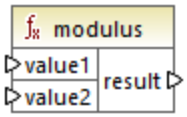
Argumento	Descripción
value	Parámetro obligatorio. Suministra el valor de entrada de la función.

Ejemplo

P. ej. si el resultado de una función de división es **11.7** y a este resultado le aplicamos la función `floor`, el resultado se convierte en **11** (es decir, el número entero menor más cercano).

6.6.6.5 modulus

El resultado es el entero restante después de dividir **value1** por **value2**.



Lenguajes

Built-in, C++, C#, Java, XQuery#, XSLT 1.0 y, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Parámetros

Argumento	Descripción
value1	Parámetro obligatorio. Suministra el primer operando
value2	Parámetro obligatorio. Suministra el segundo operando

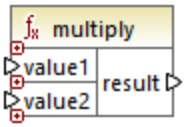
Ejemplo

Si los valores de entrada son **1.5** y **1**, entonces el resultado de la función `modulus` es **0.5**, que es la cifra que queda al dividir `1.5 / 1`.

Si los valores de entrada son **9** y **3**, entonces el resultado es **0** porque al dividir `9 / 3` no queda ningún resto.

6.6.6.6 multiply

El resultado es el valor decimal resultante de multiplicar **value1** por **value2**.



Lenguajes

Built-in, C++, C#, Java, XQuery#, XSLT 1.0 y, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Parámetros

Argumento	Descripción
value1	Parámetro obligatorio. Suministra el primer operando
value2	Parámetro obligatorio. Suministra el segundo operando

6.6.6.7 round

Devuelve el valor redondeado al número entero más cercano. Cuando el valor esté justo entre dos enteros, se utilizará el algoritmo Desempate la mitad alejándose del cero. Por ejemplo, el valor 10.5 se redondearía hasta 11 y el valor -10.5 se redondearía hasta -10.

Lenguajes

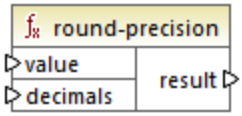
Built-in, C++, C#, Java, XQuery#, XSLT 1.0 y, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Parámetros

Argumento	Descripción
value	Parámetro obligatorio. Suministra el valor de entrada de la función.

6.6.6.8 round-precision

El resultado es el valor decimal del número redondeado a las cifras decimales definidas por el parámetro **decimals**.



Lenguajes

Disponible para Java, C#, C++ y el motor de ejecución integrado.

Parámetros

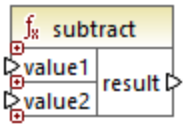
Argumento	Descripción
value	Parámetro obligatorio. Suministra el valor de entrada de la función.
decimals	Parámetro obligatorio. Indica el número de decimales a los que se debe redondear.

Ejemplo

Si redondeamos el valor **2.777777** a dos decimales el resultado es **2.78**. Si redondeamos el valor **0.1234** a 3 decimales el resultado es **0.123**.

6.6.6.9 subtract

El resultado es el valor decimal resultante de restar **value2** a **value1**.



Lenguajes

Built-in, C++, C#, Java, XQuery#, XSLT 1.0 y, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Parámetros

Argumento	Descripción
value1	Parámetro obligatorio. Suministra el primer operando
value2	Parámetro obligatorio. Suministra el segundo operando

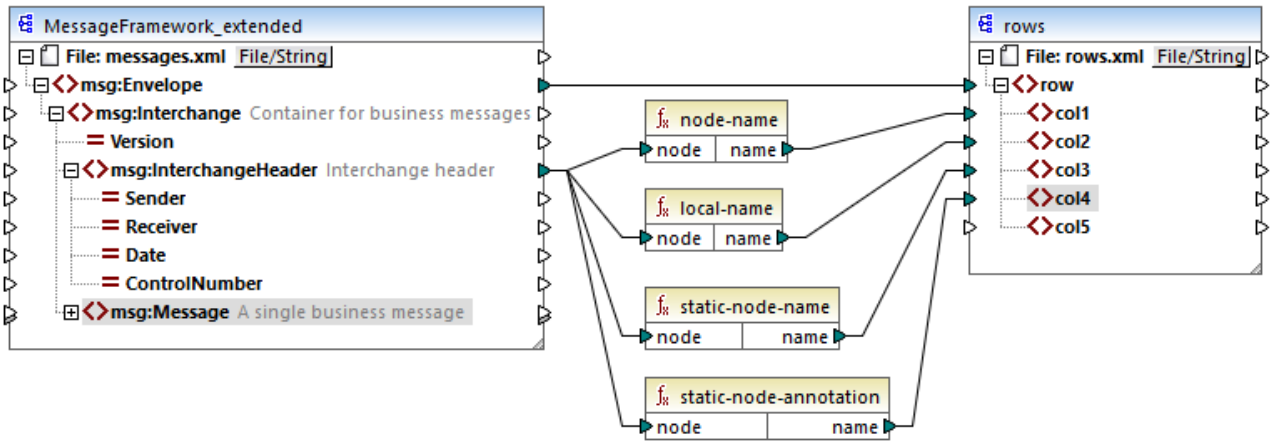
6.6.7 core | node functions (nodo)

Las funciones de la biblioteca **core | node functions** permiten acceder a información sobre los nodos de un componente de asignación (como el nombre o la anotación del nodo) o elementos nillable (véase también Valores Nil y Nillable) 121.

Tenga en cuenta que hay otra manera de acceder a los nombres de los nodos que no necesita funciones de nodo (véase Asignar nombres de nodos) 385.

La asignación siguiente muestra algunas funciones de nodo que obtienen información del nodo **msg:InterchangeHeader** del archivo XML de origen. Más concretamente, esta es la información que se extrae:

1. La función **node-name** devuelve el nombre completo del nodo, que incluye el prefijo del nodo.
2. La función **local-name** devuelve solamente la parte local.
3. La función **static-node-name** es parecida a **node-name**, pero también está disponible en XSLT 1.0.
4. La función **static-node-annotation** obtiene la anotación del elemento tal y como está definida en el esquema XML.

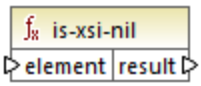


El resultado de la asignación es el siguiente (sin incluir las declaraciones XML y de espacio de nombres):


```
<row>
 <col1>msg:InterchangeHeader</col1>
 <col2>InterchangeHeader</col2>
 <col3>msg:InterchangeHeader</col3>
 <col4>Interchange header</col4>
</row>
```

6.6.7.1 is-xsi-nil

Devuelve **true** si el nodo **element** del componente de origen tiene un atributo `xsi:nil` con el valor **true**.



Lenguajes

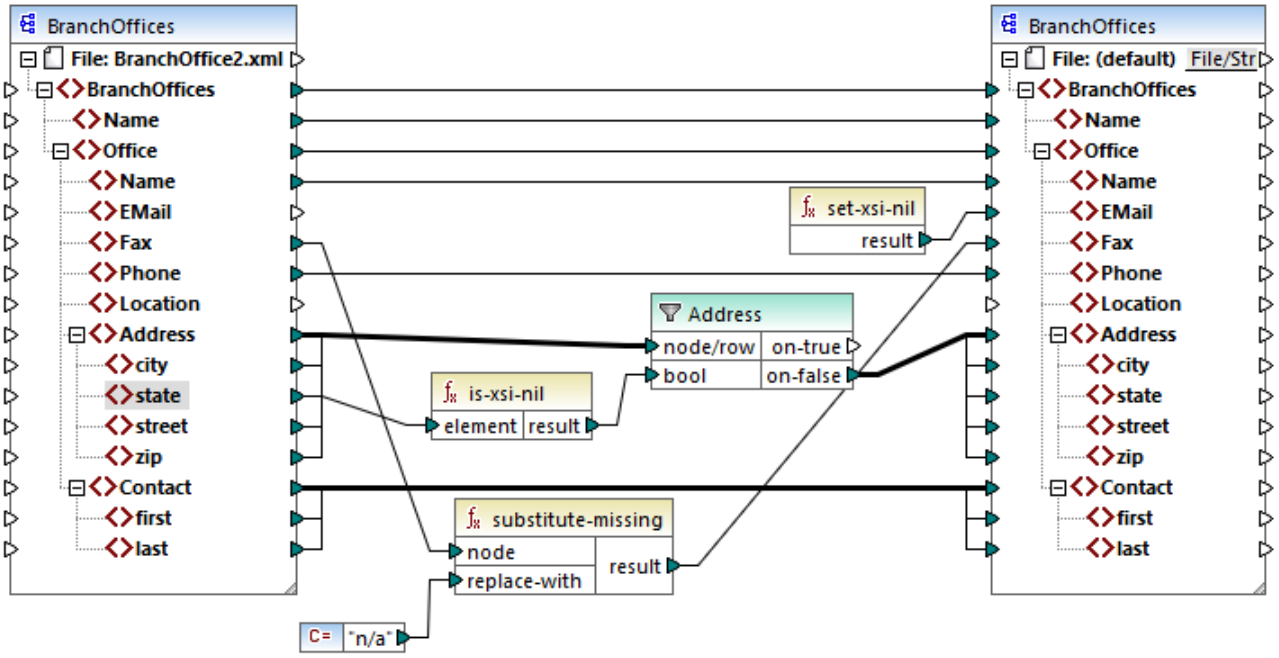
Built-in, C++, C#, Java, XQuery#, XSLT 1.0 y, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Parámetros

Argumento	Descripción
element	Parámetro obligatorio. Debe estar conectado al nodo de origen que se quiere comprobar.

Ejemplo

El diseño de asignación siguiente copia datos de un archivo XML de origen a uno de destino de forma condicional e ilustra también cómo se usan distintas funciones, incluida **`is-xsi-nil`**. Esta asignación se llama **HandlingXsiNil.mfd** y está en la carpeta **<Documentos>\Altova\MapForce2024\MapForceExamples**.



Como se ve en la imagen anterior, la función **is-xsi-nil** comprueba si el atributo `xsi:nil` tiene el atributo "true" para el elemento **state** en el archivo de origen Si, por el contrario, este atributo es "false", el filtro copiará al elemento superior **Address** en el archivo de destino. El archivo XML de origen tiene este aspecto (sin incluir las declaraciones XML y de espacio de nombres):

```
<BranchOffices>
 <Name>Nanonull</Name>
 <Office>
 <Name>Nanonull Research Outpost</Name>
 <EMail>sp@nanonull.com</EMail>
 <Fax xsi:nil="true"/>
 <Phone>+8817 3141 5926</Phone>
 <Address>
 <city>South Pole</city>
 <state xsi:nil="true"/>
 <street xsi:nil="true"/>
 <zip xsi:nil="true"/>
 </Address>
 <Contact>
 <first>Scott</first>
 <last>Amundsen</last>
 </Contact>
 </Office>
</BranchOffices>
```

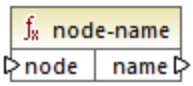
El resultado de la asignación es que no se copia ningún elemento **Address** en el archivo de destino porque sólo hay uno en el archivo de origen y el atributo `xsi:nil` es "true" para el elemento **state**. En consecuencia, el resultado de la asignación es este:

```
<BranchOffices>
 <Name>Nanonull</Name>
 <Office>
 <Name>Nanonull Research Outpost</Name>
 <EMail xsi:nil="true"/>
 <Fax>n/a</Fax>
 <Phone>+8817 3141 5926</Phone>
 <Contact>
 <first>Scott</first>
 <last>Amundsen</last>
 </Contact>
 </Office>
</BranchOffices>
```

6.6.7.2 node-name

Devuelve una cadena que contiene el nombre (QName) del nodo conectado. Si el nodo es un nodo XML **text()**, devuelve un QName vacío. Esta función solamente funciona con los nodos que tienen nombre. Si XSLT 2.0 es el lenguaje de destino (que llama a la función **fn:node-name**), la función devuelve una secuencia vacía para los nodos que no tienen nombre.

Nota: Esta función no es compatible con nodos "Introducción de archivos", tablas o campos de BD, XBRL, Excel, JSON o campos de Protocol Buffers.



Lenguajes

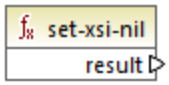
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

Parámetros

Argumento	Descripción
nodo	Parámetro obligatorio. Conecta la entrada con el nodo cuyo nombre se quiere obtener.

6.6.7.3 set-xsi-nil

Establece el nodo de destino en xsi:nil.



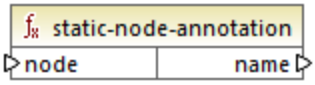
Lenguajes

Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

6.6.7.4 static-node-annotation

Devuelve una cadena con la anotación del nodo conectado. El parámetro de entrada node debe ser: (i) un nodo del componente de origen o (ii) una función inline[208] directamente conectada a un parámetro[211], directamente conectado a su vez a un nodo de la asignación que llama a la función.

La conexión debe ser directa. No puede pasar por un filtro ni por una función definida por el usuario no inline. Se trata de una pseudo función, que en tiempo de generación se reemplaza con un texto adquirido del nodo conectado y, por tanto, está disponible en todos los lenguajes.



Lenguajes

Built-in, C++, C#, Java, XQuery#, XSLT 1.0 y, XSLT 1.0, XSLT 2.0, XSLT 3.0.

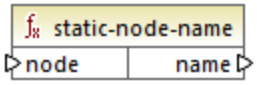
Parámetros

Argumento	Descripción
nodo	Parámetro obligatorio. Conecta el valor de entrada con el nodo cuyo anotación se quiere obtener.

6.6.7.5 static-node-name

Devuelve una cadena que contiene el nombre del nodo conectado. El parámetro de entrada node debe ser: (i) un nodo del componente de origen o (ii) una función inline[208] directamente conectada a un parámetro[211], directamente conectado a su vez a un nodo de la asignación que llama a la función.

La conexión debe ser directa. No puede pasar por un filtro ni por una función definida por el usuario no inline. Se trata de una pseudo función, que en tiempo de generación se reemplaza con un texto adquirido del nodo conectado y, por tanto, está disponible en todos los lenguajes.



Lenguajes

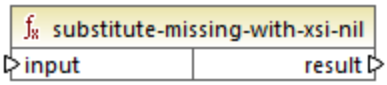
Built-in, C++, C#, Java, XQuery#, XSLT 1.0 y, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Parámetros

Argumento	Descripción
nodo	Parámetro obligatorio. Conecta la entrada con el nodo cuyo nombre se quiere obtener.

6.6.7.6 substitute-missing-with-xsi-nil

Para los nodos de contenido simple esta función reemplaza los valores que faltan o los valores nulos del componente de origen con el atributo `xsi:nil` en el nodo de destino.



Lenguajes

Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

Parámetros

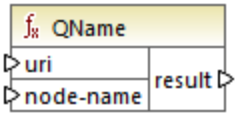
Argumento	Descripción
input	Parámetro obligatorio. Conecta la entrada con el nodo cuyo nombre se quiere obtener.

6.6.8 core | QName functions (QName)

Las funciones QName permiten manipular los nombres completos (QName) en documentos XML.

6.6.8.1 QName

Construye un QName a partir de un URI de espacio de nombres y una parte local. Use esta función para crear un QName en un componente de destino. Los parámetros **uri** y **node-name** se pueden dar con una función constante.



Lenguajes

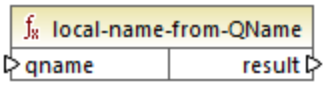
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

Parámetros

Nombre	Descripción
uri	Obligatorio. Suministra el URI.
node-name	Obligatorio. Suministra el nombre del nodo.

6.6.8.2 local-name-from-QName

Extrae la parte del nombre local de un valor de tipo `xs:QName`. Tenga en cuenta que, a diferencia de la función local-name que devuelve el nombre local del nodo, esta función procesa el contenido del elemento conectado a la entrada qnameTenga en, que devuelve el nombre local del *nodo*, esta función procesa el *contenido* del elemento conectado a la entrada **qname**



Lenguajes

Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

Parámetros

Nombre	Descripción
qname	Obligatorio. Suministra el valor de entrada de la función, que es de tipo `xs:QName`.

6.6.8.3 namespace-uri-from-QName

El resultado es la parte URI del espacio de nombres del valor QName dado como argumento.

Lenguajes

Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

Parámetros

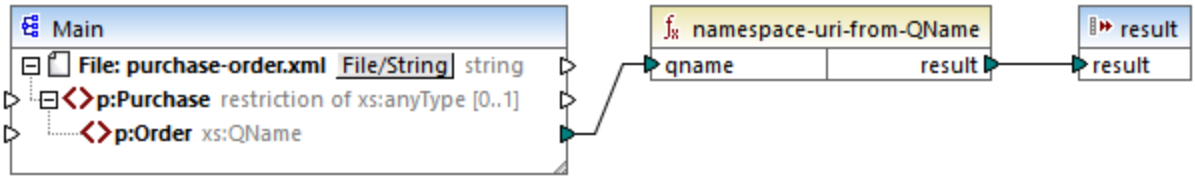
Nombre	Descripción
qname	Obligatorio. Suministra el valor de entrada de la función.

Ejemplo

Este archivo contiene el valor QName **o:name**. Observe que el prefijo "o" está asignado al espacio de nombres **`http://NamespaceTest.com/Order`**.

```
<?xml version="1.0" encoding="utf-8"?>
<p:Purchase xsi:schemaLocation="http://NamespaceTest.com/Purchase Main.xsd"
 xmlns:p="http://NamespaceTest.com/Purchase"
 xmlns:o="http://NamespaceTest.com/Order"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <p:Order>o:name</p:Order>
</p:Purchase>
```

Esta asignación procesa el valor Qname y obtiene el URI de espacio de nombre:



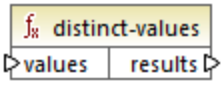
El resultado de esta asignación es **`http://NamespaceTest.com/Order`**.

6.6.9 core | sequence functions (secuencia)

Las funciones de secuencia[408] permiten procesar los datos de entrada y agrupar su contenido.

6.6.9.1 distinct-values

Procesa la secuencia de valores conectada al componente **valores** y devuelve solamente los valores distintos en forma de secuencia. Esto puede ser útil si necesita eliminar valores duplicados de una secuencia y copiar solamente los elementos únicos en el componente de destino.



Lenguajes

Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

Parámetros

Nombre	Descripción
values	Esta entrada debe recibir una conexión desde un elemento de la asignación que suministre una secuencia[408] de cero o más valores. Por ejemplo, la conexión puede provenir de un elemento XML de origen.

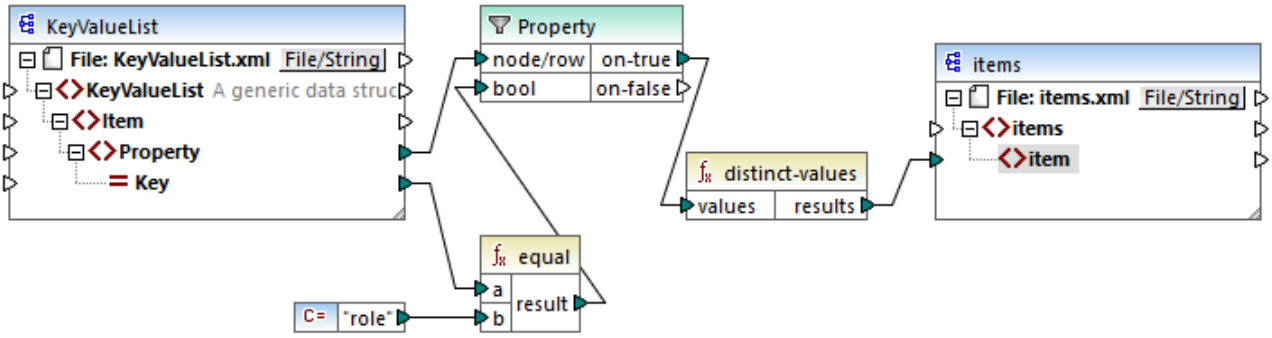
Ejemplo

Este archivo XML contiene información sobre los trabajadores de una empresa de muestra. Algunos trabajadores tienen el mismo puesto, por tanto, el atributo "role" contiene valores duplicados. Por ejemplo, tanto "Loby Matise" como "Susi Sanna" tienen el puesto "Support".

```
<?xml version="1.0" encoding="UTF-8"?>
<KeyValueList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="KeyValueList.xsd">
 <Item>
 <Property Key="role">Manager</Property>
 <Property Key="First">Vernon</Property>
 <Property Key="Last">Callaby</Property>
 </Item>
 <Item>
 <Property Key="role">Programmer</Property>
 <Property Key="First">Frank</Property>
 <Property Key="Last">Further</Property>
 </Item>
 <Item>
 <Property Key="role">Support</Property>
 <Property Key="First">Loby</Property>
 <Property Key="Last">Matise</Property>
 </Item>
 <Item>
 <Property Key="role">Support</Property>
 <Property Key="First">Susi</Property>
```

```
 <Property Key="Last">Sanna</Property>
 </Item>
</KeyValueList>
```

Imagine que necesita extraer una lista de todos los nombres de puesto únicos del archivo XML. Para ello puede usar una asignación como esta:



En la asignación anterior ocurre lo siguiente:

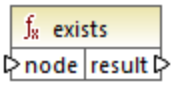
- Cada elemento **Property** del archivo XML de origen es procesado por un filtro.
- La conexión con el elemento de entrada **bool** del filtro garantiza que sólo se pasen al componente de destino elementos **Property** en los que el atributo **Key** es igual a "role". La cadena "role" viene dada por una constante. Tenga en cuenta que el resultado del filtro todavía produce duplicados de momento (porque hay dos propiedades "Support" que cumplen la condición del filtro).
- La secuencia que produce el filtro es procesada por la función **distinct-values**, que excluye los valores duplicados.

El resultado de la asignación es el siguiente:

```
<items>
 <item>Manager</item>
 <item>Programmer</item>
 <item>Support</item>
</items>
```

6.6.9.2 exists

Devuelve **true** si el nodo existe. De lo contrario devuelve **false**. Esta función se suele usar con filtros porque devuelve un valor booleano, lo que permite filtrar [178] por registros que tienen (o que no tienen) un elemento o atributo secundario.



Lenguajes

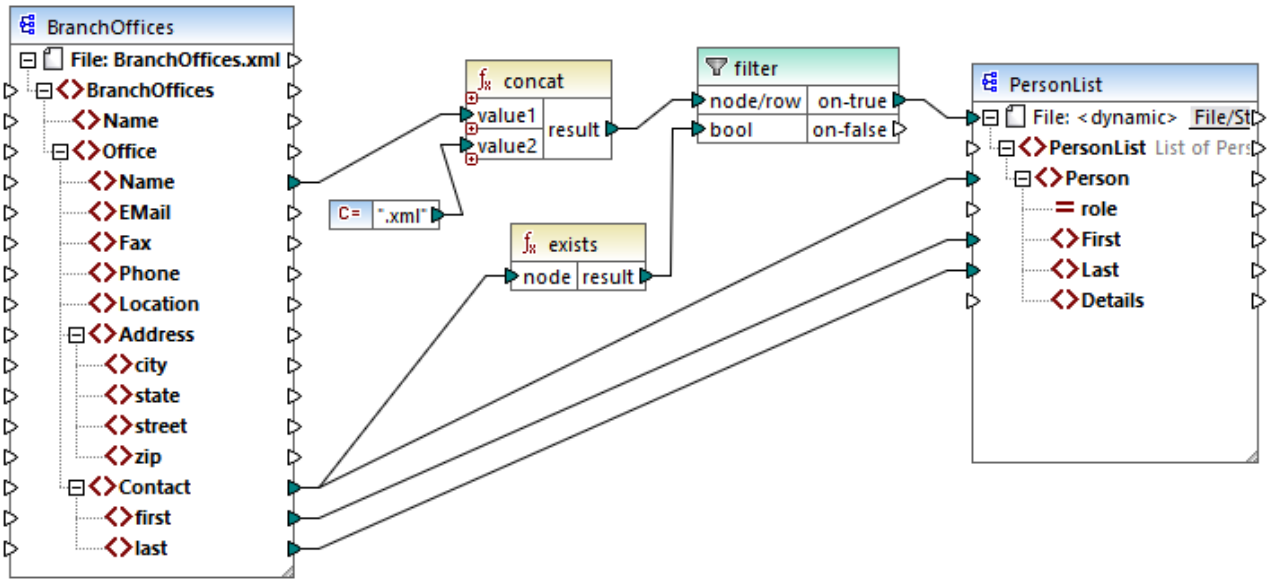
Built-in, C++, C#, Java, XQuery#, XSLT 1.0 y, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Parámetros

Nombre	Descripción
nodo	El nodo cuya existencia se quiere comprobar.

Ejemplos

La asignación siguiente muestra cómo filtrar datos con ayuda de la función **exists**. Esta asignación se llama **PersonListsForAllBranchOffices.mfd** y está en la carpeta **<Documentos>\Altova\MapForce2024\MapForceExamples**.



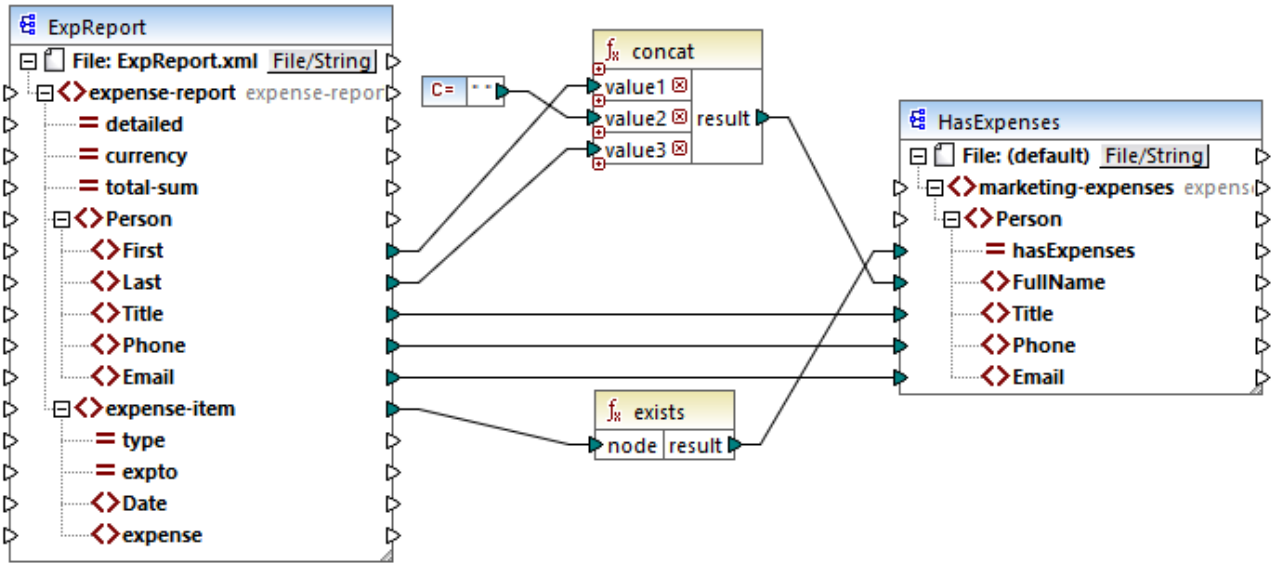
PersonListsForAllBranchOffices.mfd

En el archivo de origen **BranchOffices.xml** hay tres elementos **Office**. En concreto, uno de ellos no tienen ningún elemento secundario **Contact**. La asignación tiene varios objetivos:

a) por cada oficina, extraer la lista de contactos de esa oficina
b) por cada oficina, crear un archivo XML con el mismo nombre que esa oficina
C) no generar el archivo XML si la oficina no tiene contactos.

Para ello se ha añadido un filtro a la asignación. El filtro pasa al archivo de destino solamente los elementos **Office** en los que existe al menos un elemento **Contact**. Esta condición booleana viene dada por la función **exists**. Si el resultado de la función es true, entonces el nombre de la oficina se concatena con la cadena de texto `.xml` para producir el nombre del archivo de destino. Para más información sobre cómo generar nombres de archivo para la asignación consulte Procesar varios archivos de entrada o salida simultáneamente 402 .

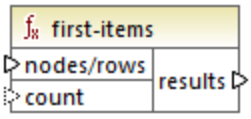
Otro ejemplo sería esta asignación:
<Documentos>\Altova\MapForce2024\MapForceExamples\HasMarketingExpenses.mfd. Si existe un elemento **expense-item** en el XML de origen, entonces el atributo "hasExpenses" recibe el valor **true** en el esquema o archivo XML de destino.



HasMarketingExpenses.mfd

6.6.9.3 first-items

Devuelve los *X* últimos nodos de la secuencia nodes/rows, siendo *X* el número dado por el parámetro **count**.



Lenguajes

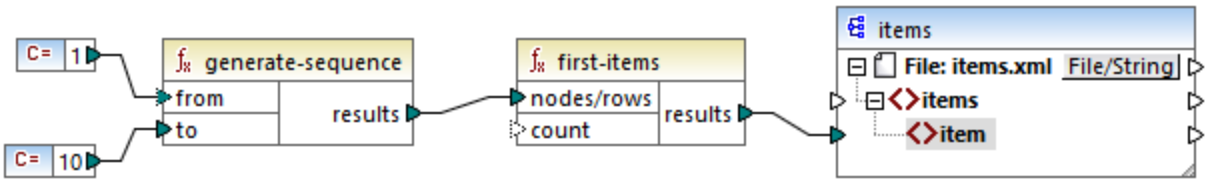
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

Parámetros

Nombre	Descripción
nodos/filas	Esta entrada debe recibir una conexión desde un elemento de la asignación que suministre una secuencia[408] de cero o más valores. Por ejemplo, la conexión puede provenir de un elemento XML de origen.
count	Parámetro opcional. Indica cuántos nodos se deben recuperar de la secuencia de entrada. El valor predeterminado es **1**.

Ejemplo

La asignación de ejemplo siguiente genera una secuencia de 10 valores. Esta secuencia es procesada por la función **first-items**, que escribe el resultado en un archivo XML de destino.



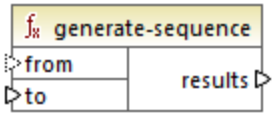
Como el argumento **count** no tiene ningún valor, se aplica el valor predeterminado **1**. El resultado es que en la salida de la asignación sólo se genera el primer valor de la secuencia:

```
<items>
 <item>1</item>
</items>
```

Para ver un ejemplo más realista consulte la asignación **FindHighestTemperatures.mfd** de Pasar parámetros a la asignación 147.

6.6.9.4 generate-sequence

Crea una secuencia de enteros usando como límite los parámetros from y to.



Lenguajes

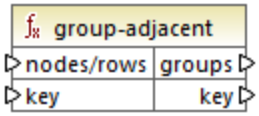
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

Parámetros

Nombre	Descripción
from	Parámetro opcional. Indica el número entero con el que debe empezar la secuencia (límite inferior). El valor predeterminado es **1**.
to	Parámetro obligatorio. Indica el número entero con el que debe terminar la secuencia (límite superior).

6.6.9.5 group-adjacent

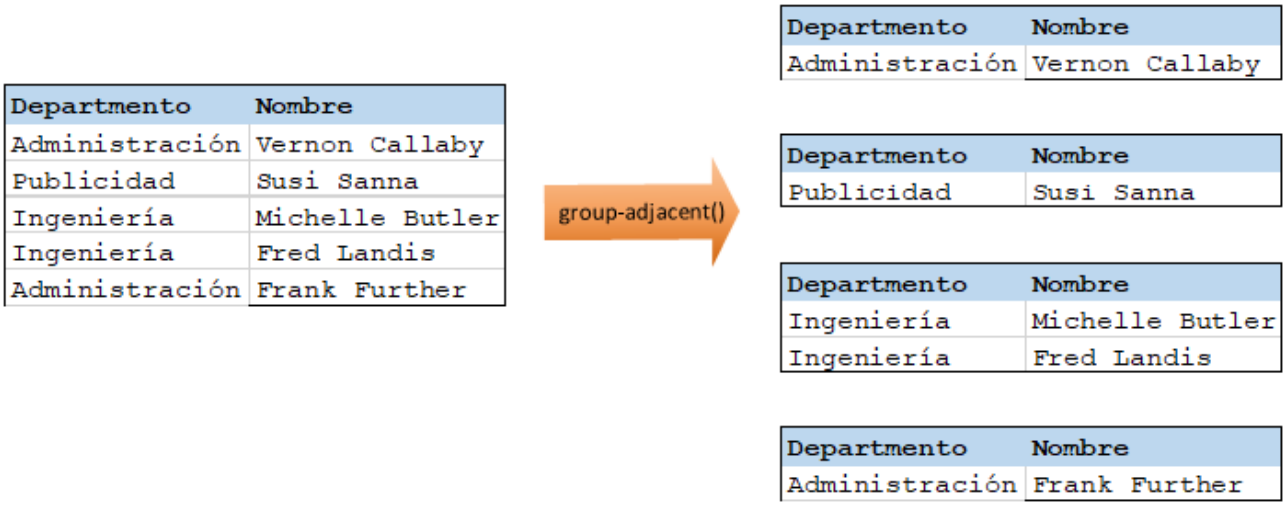
La función **group-adjacent** agrupa los elementos conectados a los datos de entrada de **nodes/rows** con la clave conectada a los datos de entrada de **key**. Observe que si los elementos que comparten la misma clave no son adyacentes, esta clave los coloca en grupos separados. Si varios elementos consecutivos (adyacentes) comparten la misma clave estarán en el mismo grupo.



Por ejemplo, en la imagen siguiente, que ilustra una transformación abstracta, la clave de agrupación es "Department". El lado izquierdo del diagrama muestra los datos de entrada, mientras que el lado derecho muestra los datos de salida tras la agrupación. Al ejecutar la transformación ocurre lo siguiente:

- Inicialmente, la primera clave, "Administración", crea un grupo nuevo.
- La siguiente clave es distinta, así que se crea un segundo grupo, "Publicidad".
- Como la tercera clave también es diferente, se crea otro grupo, "Ingeniería".
- La cuarta clave es la misma que la tercera, por lo que este registro se coloca en el grupo que ya existe para esa clave.
- Por último, la quinta clave es diferente de la cuarta, lo que hace que se cree un último grupo.

Como se ve en la siguiente imagen, "Michelle Butler" y "Fred Landis" están en el mismo grupo porque tienen la misma clave y son adyacentes. Sin embargo, "Vernon Callaby" y "Frank Further" están en grupos distintos porque no son adyacentes, aunque tengan la misma clave.



Lenguajes

La función está disponible para XSLT 2.0, XSLT 3.0, Java, C#, C++ y el motor de ejecución integrado.

Parámetros

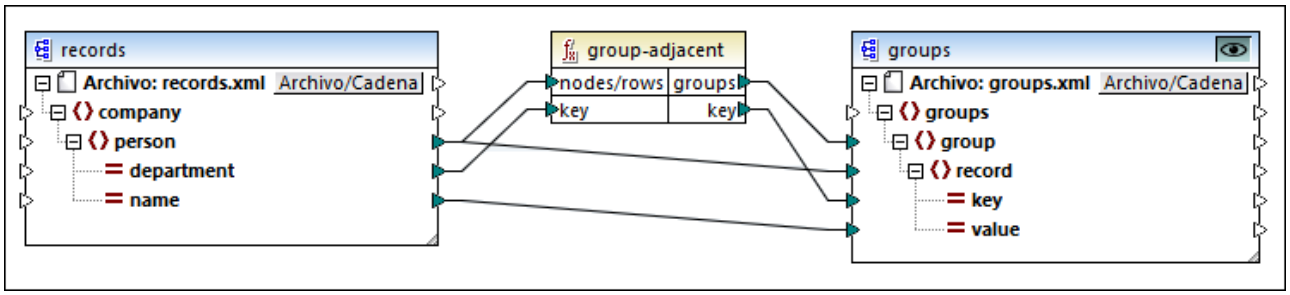
Nombre	Descripción
nodos/filas	Esta entrada debe recibir una conexión desde un elemento de la asignación que suministre una secuencia[408] de cero o más valores. Por ejemplo, la conexión puede provenir de un elemento XML de origen.
key	La clave conforme a la cual se agrupan los elementos.

Ejemplo

Imaginemos que sus datos de origen son un archivo XML con el siguiente contenido (tenga en cuenta que, para simplificar, en el código de ejemplo siguiente hemos eliminado el espacio de nombre y las declaraciones XML).

```
<company>
 <person department="Administration" name="Vernon Callaby"/>
 <person department="Marketing" name="Susi Sanna"/>
 <person department="Engineering" name="Michelle Butler"/>
 <person department="Engineering" name="Fred Landis"/>
 <person department="Administration" name="Frank Further"/>
</company>
```

El requisito empresarial es agrupar registros de personas por departamento, siempre que sean adyacentes. Para conseguirlo, la siguiente asignación invoca la función **group-adjacent** y da como clave el elemento **department**.



El resultado de la asignación es el siguiente:

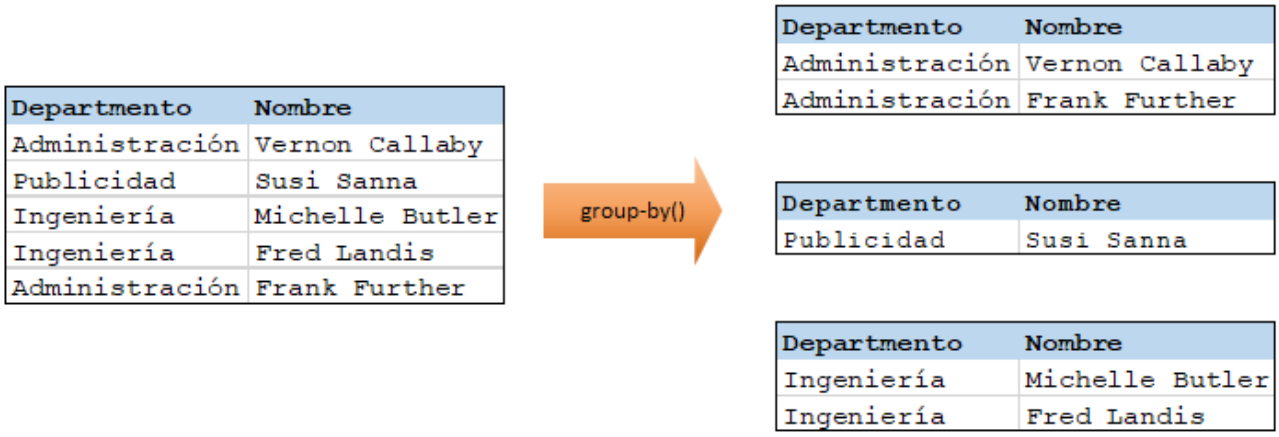
```
<groups>
 <group>
 <record key="Administration" value="Vernon Callaby"/>
 </group>
 <group>
 <record key="Marketing" value="Susi Sanna"/>
 </group>
 <group>
 <record key="Engineering" value="Michelle Butler"/>
```

```
 <record key="Engineering" value="Fred Landis"/>
 </group>
 <group>
 <record key="Administration" value="Frank Further"/>
 </group>
</groups>
```

Este ejemplo, junto con otros ejemplos de agrupación, es parte de esta asignación: **<Documentos>\Altova\MapForce2024\MapForceExamples\Tutorial\GroupingFunctions.mfd**. Antes de hacer clic en la pestaña **Resultados** recuerde hacer clic en el botón **Vista previa** de la función que quiere comprobar.

6.6.9.6 group-by

La función **group-by** crea grupos de registros conforme a las claves de agrupación que se indiquen. Por ejemplo, en el siguiente ejemplo de transformación abstracta la clave de agrupación es "Departmento". Como hay un total de tres departamentos únicos, al aplicar la función **group-by** se crean tres grupos:



Lenguajes

Built-in, C++, C#, Java, XSLT 2.0, XSLT 3.0.

Parámetros

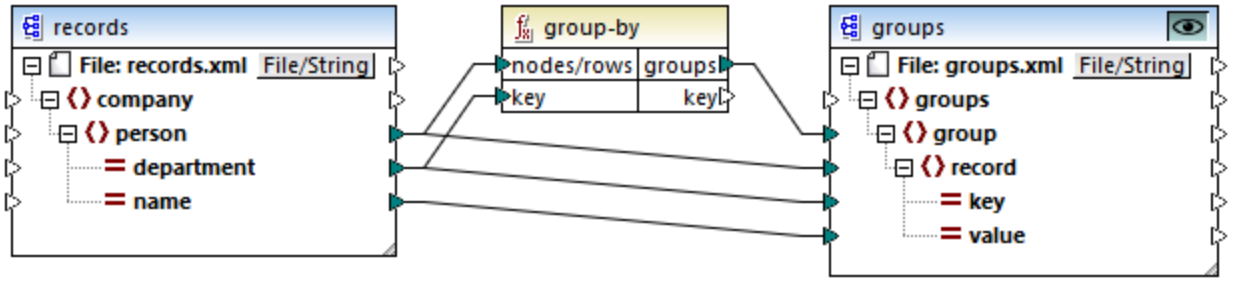
Nombre	Descripción
nodos/filas	Esta entrada debe recibir una conexión desde un elemento de la asignación que suministre una secuencia[408] de cero o más valores. Por ejemplo, la conexión puede provenir de un elemento XML de origen.
key	La clave conforme a la cual se agrupan los elementos.

Ejemplo 1

Imaginemos que sus datos de origen son un archivo XML con el siguiente contenido (tenga en cuenta que, para simplificar, en el código de ejemplo siguiente hemos eliminado el espacio de nombre y las declaraciones XML).

```
<company>
   <person department="Administration" name="Vernon Callaby"/>
   <person department="Marketing" name="Susi Sanna"/>
   <person department="Engineering" name="Michelle Butler"/>
   <person department="Engineering" name="Fred Landis"/>
   <person department="Administration" name="Frank Further"/>
</company>
```

El requisito empresarial es agrupar registros de personas por departamento. Para conseguirlo, la siguiente asignación invoca la función **group-by** y da como clave el elemento **department**.



El resultado de la asignación es el siguiente:

```
<groups>
   <group>
      <record key="Administration" value="Vernon Callaby"/>
      <record key="Administration" value="Frank Further"/>
   </group>
   <group>
      <record key="Marketing" value="Susi Sanna"/>
   </group>
   <group>
      <record key="Engineering" value="Michelle Butler"/>
      <record key="Engineering" value="Fred Landis"/>
   </group>
</groups>
```

Este ejemplo, junto con otros ejemplos de agrupación, es parte de esta asignación: **<Documentos>\Altova\MapForce2024\MapForceExamples\Tutorial\GroupingFunctions.mfd**. Antes de hacer clic en la pestaña **Resultados**, recuerde hacer clic en el botón **Vista previa** de la función que quiere comprobar.

Ejemplo 2

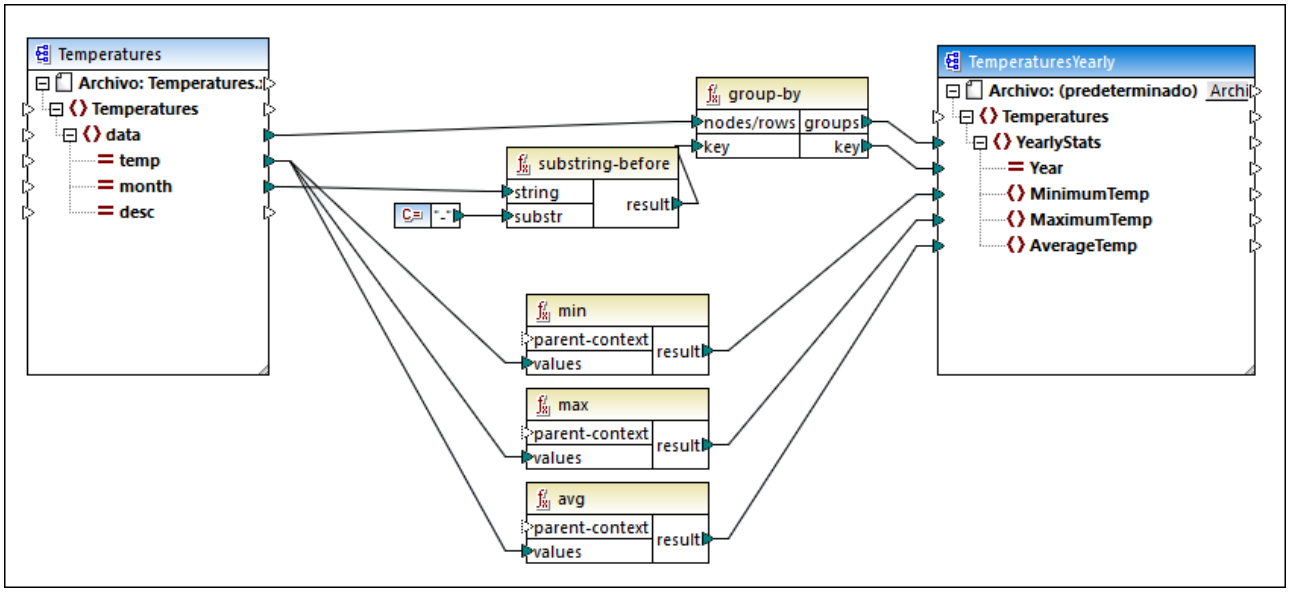
Este ejemplo muestra cómo agrupar registros con la ayuda de la función **group-by** e ilustra cómo añadir datos. Este ejemplo viene acompañado por una asignación de muestra que encontrará en la siguiente ruta: **<Documentos>\Altova\MapForce2024\MapForceExamples\GroupTemperaturesByYear.mfd**. Esta asignación lee datos de un archivo XML que contiene un registro de temperaturas mensuales, como se puede ver en el siguiente código de ejemplo:

```
<Temperatures>
   <data temp="-3.6" month="2006-01" />
   <data temp="-0.7" month="2006-02" />
   <data temp="7.5" month="2006-03" />
   <data temp="12.4" month="2006-04" />
   <data temp="16.2" month="2006-05" />
   <data temp="19" month="2006-06" />
   <data temp="22.7" month="2006-07" />
   <data temp="23.2" month="2006-08" />
   <data temp="18.7" month="2006-09" />
   <data temp="11.2" month="2006-10" />
   <data temp="9.1" month="2006-11" />
   <data temp="0.8" month="2006-12" />
   <data temp="-3.2" month="2007-01" />
   <data temp="-0.3" month="2007-02" />
   <data temp="6.5" month="2007-03" />
   <data temp="10.6" month="2007-04" />
   <data temp="19" month="2007-05" />
   <data temp="20.3" month="2007-06" />
   <data temp="22.3" month="2007-07" />
   <data temp="20.7" month="2007-08" />
   <data temp="19.2" month="2007-09" />
   <data temp="12.9" month="2007-10" />
   <data temp="8.1" month="2007-11" />
   <data temp="1.9" month="2007-12" />
</Temperatures>
```

El requisito empresarial de esta asignación es doble:

1. Agrupar todas las temperaturas de cada año.
2. Averiguar las temperaturas mínima, máxima y media de cada año.

Para conseguir el primer requisito, usamos la función **group-by**. Para conseguir el segundo requisito, usamos funciones agregadas **min** 241, **max** 240 y **avg** 238.



GroupTemperaturesByYear.mfd

La manera de ejecutar asignaciones de datos con MapForce Server (y el método recomendado para empezar a leerlas) consiste en observar el elemento de mayor nivel del componente de destino. En este ejemplo se creará un elemento **YearlyStats** por cada grupo que devuelva la función **group-by**. La función **group-by** toma como primer argumento todos los elementos **data** del origen y los agrupa en función de las conexiones que existan con la entrada de **key**. El requisito es agrupar las temperaturas por año, por lo que primero debemos obtener el año. Para ello, la función **substring-before** 315 extrae la parte que contiene el año del atributo **month** de cada elemento **data**. Es decir, toma como argumento el valor de **month** y devuelve la parte que encuentre antes de la primera instancia de **substr**. Como se ilustra más arriba, en este ejemplo **substr** equivale al signo menos o guion, por lo que si tenemos el valor "2006-01", la función devolverá "2006".

Por último, los valores **MinimumTemp**, **MaximumTemp** y **AverageTemp** se obtienen conectando estos elementos con las correspondientes funciones agregadas: **min**, **max** y **avg**. Estas tres funciones toman como entrada la secuencia de temperaturas que leen en el componente de origen. Estas funciones no necesitan un argumento **parent-context** porque ya funcionan en el contexto de cada grupo. En otras palabras, la conexión que existe entre **data** y **YearlyStats** da el contexto en el que debe trabajar cada función agregada.

Para obtener una vista previa del resultado de la asignación, haga clic en la pestaña **Resultados**. Observe que el número de grupos coincide con el número de años obtenido leyendo el archivo fuente, por ejemplo:

```
<Temperatures>
   <YearlyStats Year="2006">
      <MinimumTemp>-3.6</MinimumTemp>
      <MaximumTemp>23.2</MaximumTemp>
      <AverageTemp>11.375</AverageTemp>
   </YearlyStats>
   <YearlyStats Year="2007">
      <MinimumTemp>-3.2</MinimumTemp>
      <MaximumTemp>22.3</MaximumTemp>
      <AverageTemp>11.5</AverageTemp>
```



```
   </YearlyStats>
</Temperatures>
```

Nota: Para simplificar, el código de ejemplo anterior contiene menos datos que los datos de entrada y salida utilizados en la asignación de ejemplo.

6.6.9.7 group-ending-with

La función **group-ending** toma como argumento una condición booleana. Si la condición booleana se cumple, se crea un grupo nuevo en el que el último registro es el que cumple esa condición. En el siguiente ejemplo, la condición es que la clave sea "cola". Esta condición se cumple en los registros tercero y quinto, por lo que se crean dos grupos:

Nota: si existen registros después del último que cumple la condición, se crea un grupo más. Por ejemplo, si hubiera más registros "línea" después del último registro "cola", estos se colocarían en ese grupo adicional.

Lenguajes

La función está disponible para XSLT 2.0, XSLT 3.0, Java, C#, C++ y el motor de ejecución integrado.

Parámetros

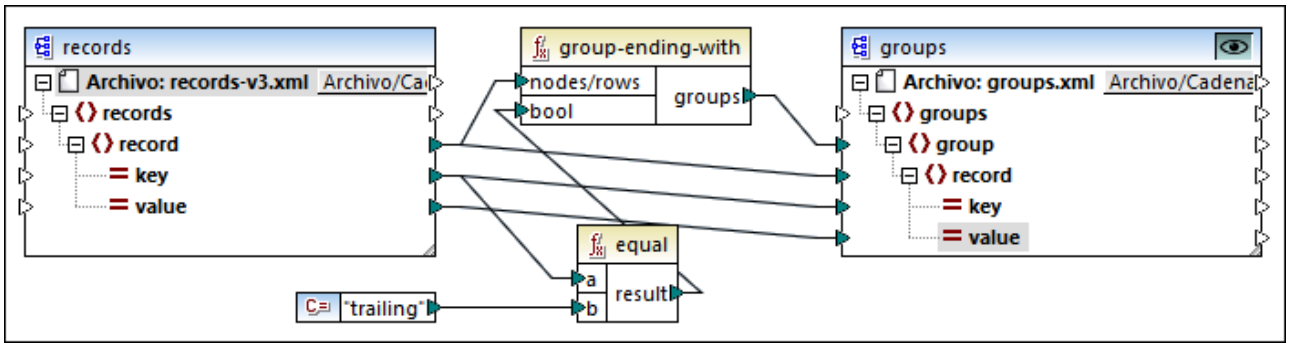
Nombre	Descripción
nodos/filas	Esta entrada debe recibir una conexión desde un elemento de la asignación que suministre una secuencia[408] de cero o más valores. Por ejemplo, la conexión puede provenir de un elemento XML de origen.
bool	xs:boolean Indica la condición booleana que inicia un nuevo grupo si se cumple (true).

Ejemplo

Imaginemos que sus datos de origen son un archivo XML con el siguiente contenido (tenga en cuenta que, para simplificar, en el código de ejemplo siguiente hemos eliminado el espacio de nombre y las declaraciones XML).

```
<records>
 <record key="line" value="A"/>
 <record key="line" value="B"/>
 <record key="trailing" value="Total 1"/>
 <record key="line" value="C"/>
 <record key="trailing" value="Total 2"/>
</records>
```

El requisito empresarial es crear un grupo por cada elemento "trailing". Cada uno de los grupos también debe incluir los registros "line" que precedan al registro "trailing". Para ello, la siguiente asignación invoca la función **group-ending-with**. En la siguiente asignación, siempre que el nombre de **key** sea "trailing", el argumento dado a **bool** pasa a ser **true** y se crea un grupo nuevo.



El resultado de la asignación es el siguiente:

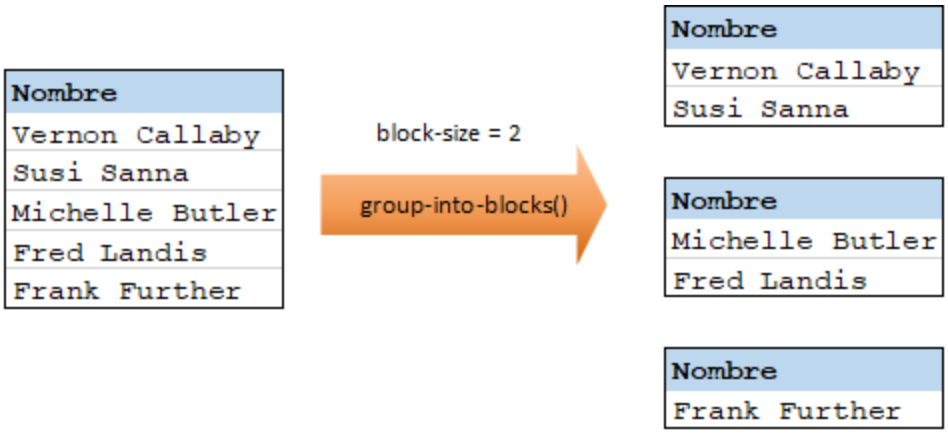
```
<groups>
 <group>
 <record key="line" value="A"/>
 <record key="line" value="B"/>
 <record key="trailing" value="Total 1"/>
 </group>
 <group>
 <record key="line" value="C"/>
 <record key="trailing" value="Total 2"/>
 </group>
</groups>
```

Este ejemplo, junto con otros ejemplos de agrupación, es parte de esta asignación: **<Documentos>\Altova\MapForce2024\MapForceExamples\Tutorial\GroupingFunctions.mfd**. Antes de hacer clic en la pestaña **Resultados** recuerde hacer clic en el botón **Vista previa** de la función que quiere comprobar.

6.6.9.8 group-into-blocks

La función **group-into-blocks** crea grupos iguales que contienen exactamente N elementos, donde N es el valor que se le da al argumento `block-size`. Observe que el último grupo puede contener N elementos o

menos en función de la cantidad de elementos de la fuente. En el ejemplo siguiente, `block-size` es 2. Como hay cinco elementos en total, cada grupo contiene exactamente dos elementos, excepto el último, que sólo puede contener uno.



Lenguajes

La función está disponible para XSLT 2.0, XSLT 3.0, Java, C#, C++ y el motor de ejecución integrado.

Parámetros

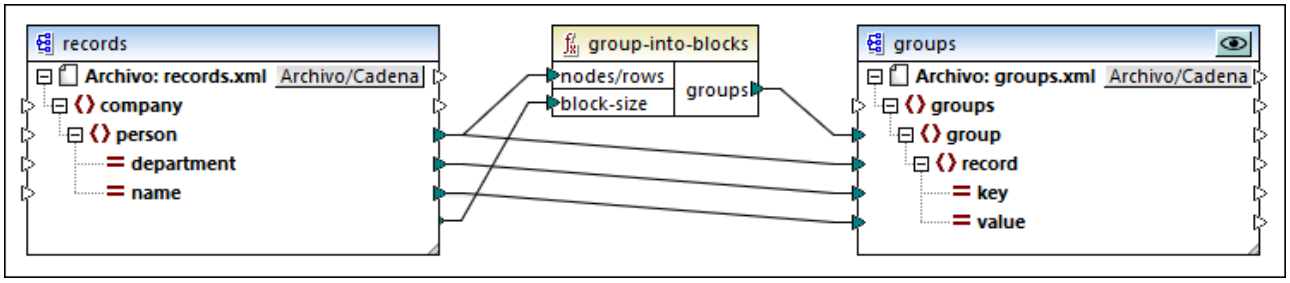
Nombre	Descripción
nodos/filas	Esta entrada debe recibir una conexión desde un elemento de la asignación que suministre una secuencia[408] de cero o más valores. Por ejemplo, la conexión puede provenir de un elemento XML de origen.
block-size	Indica el tamaño de cada grupo.

Ejemplo

Imaginemos que sus datos de origen son un archivo XML con el siguiente contenido (tenga en cuenta que, para simplificar, en el código de ejemplo siguiente hemos eliminado el espacio de nombre y las declaraciones XML).

```
<company>
 <person department="Administration" name="Vernon Callaby"/>
 <person department="Marketing" name="Susi Sanna"/>
 <person department="Engineering" name="Michelle Butler"/>
 <person department="Engineering" name="Fred Landis"/>
 <person department="Administration" name="Frank Further"/>
</company>
```

El requisito empresarial es agrupar registros de personas en bloques de dos elementos cada uno. Para ello, la siguiente asignación invoca la función **`group-into-blocks`** y da como tamaño (**block-size**) el valor entero "2".



El resultado de la asignación es el siguiente:

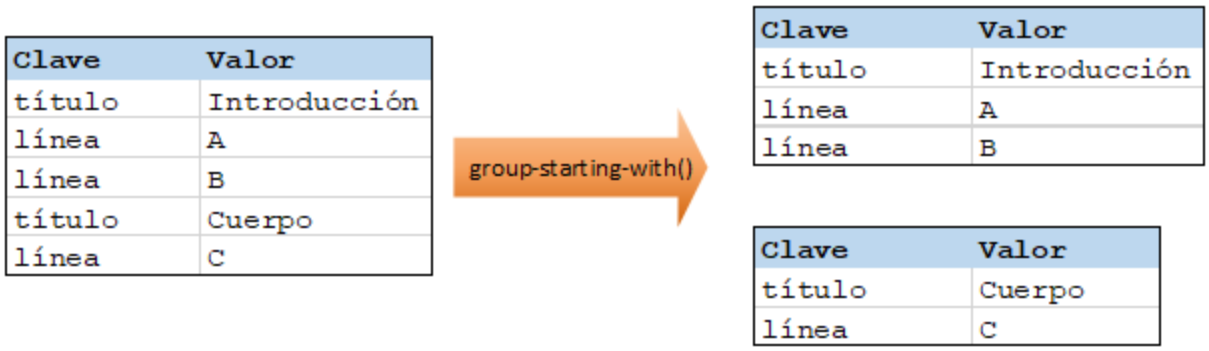
```
<groups>
 <group>
 <record key="Administration" value="Vernon Callaby"/>
 <record key="Marketing" value="Susi Sanna"/>
 </group>
 <group>
 <record key="Engineering" value="Michelle Butler"/>
 <record key="Engineering" value="Fred Landis"/>
 </group>
 <group>
 <record key="Administration" value="Frank Further"/>
 </group>
</groups>
```

Observe que el último grupo contiene un único elemento, ya que el número total de elementos (5) no se puede dividir entre 2.

Este ejemplo, junto con otros ejemplos de agrupación, es parte de esta asignación: **<Documentos>\Altova\MapForce2024\MapForceExamples\Tutorial\GroupingFunctions.mfd**. Antes de hacer clic en la pestaña **Resultados** recuerde hacer clic en el botón **Vista previa** de la función que quiere comprobar.

6.6.9.9 group-starting-with

La función **group-starting-with** toma como argumento una condición booleana. Si se cumple la condición booleana se crea un grupo nuevo que empieza con el registro que cumple esa condición. En el siguiente ejemplo la condición es que la clave "Clave" debe ser "título". Esta condición se cumple en el caso de los registros primero y cuarto, así que se crean dos grupos:



Nota: si existen registros antes del primero que cumple la condición, se crea un grupo más. Por ejemplo, si hubiera más registros "línea" antes del último registro "título", estos se colocarían en ese grupo adicional.

Lenguajes

La función está disponible para XSLT 2.0, XSLT 3.0, Java, C#, C++ y el motor de ejecución integrado.

Parámetros

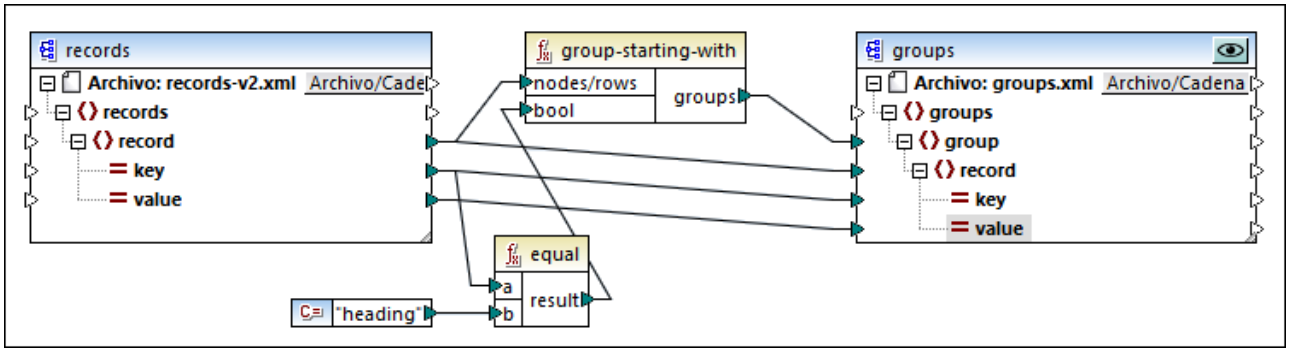
Nombre	Descripción
nodos/filas	Esta entrada debe recibir una conexión desde un elemento de la asignación que suministre una secuencia[408] de cero o más valores. Por ejemplo, la conexión puede provenir de un elemento XML de origen.
bool	xs:boolean Indica la condición booleana que inicia un nuevo grupo si se cumple (true).

Ejemplo

Imaginemos que sus datos de origen son un archivo XML con el siguiente contenido (tenga en cuenta que, para simplificar, en el código de ejemplo siguiente hemos eliminado el espacio de nombre y las declaraciones XML).

```
<records>
 <record key="heading" value="Intro"/>
 <record key="line" value="A"/>
 <record key="line" value="B"/>
 <record key="heading" value="Body"/>
 <record key="line" value="C"/>
</records>
```

El requisito empresarial es crear un grupo para cada registro "heading". Cada grupo también debe incluir los registros "line" que siguen al registro "heading". Para ello, la siguiente asignación invoca la función **group-starting-with**. En la siguiente asignación, siempre que el nombre de **key** sea "heading", el argumento dado a **bool** pasa a ser **true** y se crea un grupo nuevo.



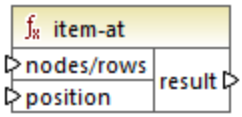
El resultado de la asignación es el siguiente:

```
<groups>
 <group>
 <record key="heading" value="Intro"/>
 <record key="line" value="A"/>
 <record key="line" value="B"/>
 </group>
 <group>
 <record key="heading" value="Body"/>
 <record key="line" value="C"/>
 </group>
</groups>
```

Este ejemplo, junto con otros ejemplos de agrupación, es parte de esta asignación: **<Documentos>\Altova\MapForce2024\MapForceExamples\Tutorial\GroupingFunctions.mfd**. Antes de hacer clic en la pestaña **Resultados** recuerde hacer clic en el botón **Vista previa** de la función que quiere comprobar.

6.6.9.10 item-at

Devuelve un elemento de la secuencia de nodos/filas dados como argumento que se encuentra en la posición indicada por el argumento **position**. El primer elemento está en la posición **1**.



Lenguajes

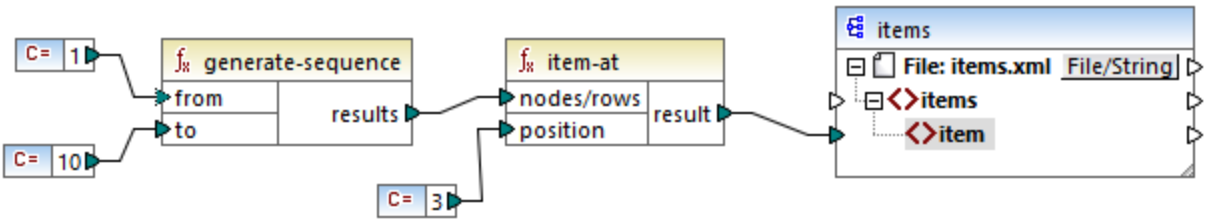
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

Parámetros

Nombre	Descripción
nodos/filas	Esta entrada debe recibir una conexión desde un elemento de la asignación que suministre una secuencia[408] de cero o más valores. Por ejemplo, la conexión puede provenir de un elemento XML de origen.
position	Este número entero indica qué elemento de la secuencia se debe devolver.

Ejemplo

La asignación de ejemplo siguiente genera una secuencia de 10 valores. Esta secuencia es procesada por la función **item-at**, que escribe el resultado en un archivo XML de destino.

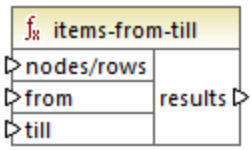


El argumento **position** tiene el valor **3**, por lo que sólo el tercer valor de la secuencia pasa al archivo de destino. En consecuencia, la salida de la asignación es la siguiente (sin incluir las declaraciones de esquema XML):

```
<items>
 <item>3</item>
</items>
```

6.6.9.11 items-from-till

Devuelve una secuencia de **nodes/rows** usando los parámetros "from" y "till" como límite de la secuencia. El primer elemento está en la posición **1**.



Lenguajes

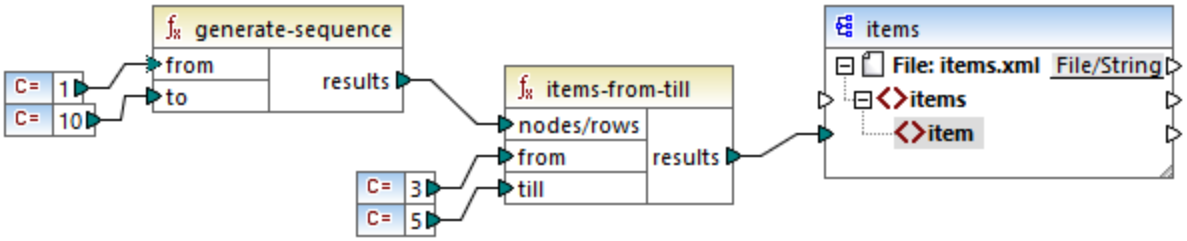
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

Parámetros

Nombre	Descripción
nodos/filas	Esta entrada debe recibir una conexión desde un elemento de la asignación que suministre una secuencia[408] de cero o más valores. Por ejemplo, la conexión puede provenir de un elemento XML de origen.
from	Este número entero indica la posición inicial desde la que se deben recuperar los elementos.
till	Este número entero indica hasta qué posición se deben recuperar elementos.

Ejemplo

La asignación de ejemplo siguiente genera una secuencia de 10 valores. Esta secuencia es procesada por la función **items-from-till**, que escribe el resultado en un archivo XML de destino.

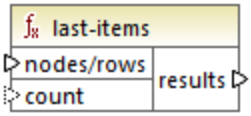


Los argumentos **from** y **till** tienen los valores **3** y **5**, por lo que sólo se pasan al componente de destino el subconjunto de valores de **3** a **5**. En consecuencia, la salida de la asignación es la siguiente (sin incluir las declaraciones de esquema XML):

```
<items>
 <item>3</item>
 <item>4</item>
 <item>5</item>
</items>
```

6.6.9.12 last-items

Devuelve los *X* últimos nodos de la secuencia nodos/filas, siendo *X* el número dado por el parámetro **count**. El primer elemento está en la posición 1.



Lenguajes

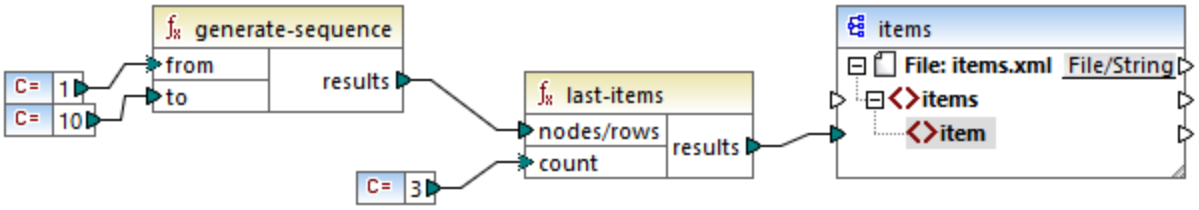
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

Parámetros

Nombre	Descripción
nodos/filas	Esta entrada debe recibir una conexión desde un elemento de la asignación que suministre una secuencia[408] de cero o más valores. Por ejemplo, la conexión puede provenir de un elemento XML de origen.
count	Parámetro opcional. Indica cuántos nodos se deben recuperar de la secuencia de entrada. El valor predeterminado es **1**.

Ejemplo

La asignación de ejemplo siguiente genera una secuencia de 10 valores. Esta secuencia es procesada por la función `last-items`, que escribe el resultado en un archivo XML de destino.

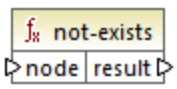


El argumento count es **3**, por lo que sólo los tres últimos valores de la secuencia se pasan al archivo de destino. En consecuencia, la salida de la asignación es la siguiente (sin incluir las declaraciones de esquema XML):

```
<items>
 <item>8</item>
 <item>9</item>
 <item>10</item>
</items>
```

6.6.9.13 not-exists

Si el nodo existe, la función devuelve **false**. De lo contrario, devuelve **true**. Esta función es la contraria a la función exists[281] pero funciona de la misma manera.



Lenguajes

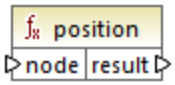
Built-in, C++, C#, Java, XQuery#, XSLT 1.0 y, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Parámetros

Nombre	Descripción
nodo	El nodo cuya existencia se quiere comprobar.

6.6.9.14 position

Devuelve la posición de un nodo dentro de la secuencia de la que forma parte. Esta función se puede usar, por ejemplo, para numerar elementos automáticamente de forma secuencial.



Lenguajes

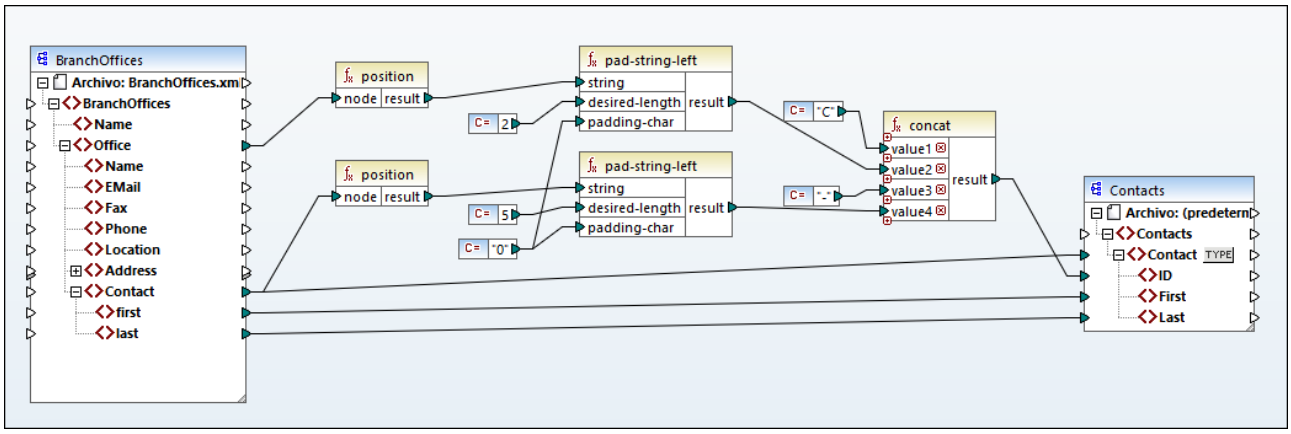
Built-in, C++, C#, Java, XQuery#, XSLT 1.0 y, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Parámetros

Nombre	Descripción
nodo	Esta entrada debe recibir una conexión desde un elemento de la asignación que suministre una secuencia 408 de cero o más valores. Por ejemplo, la conexión puede provenir de un elemento XML de origen.

Ejemplo

La función **`position`** permite determinar la posición de un nodo de una secuencia o usar una posición concreta para filtrar elementos dependiendo de su posición. Esta asignación viene acompañada por un archivo de diseño de asignación que encontrará en esta ruta:
<Documentos>\Altova\MapForce2024\MapForceExamples\ContactsFromBranchOffices.mfd.



ContactsFromBranchOffices.mfd

En la asignación anterior, el archivo XML de origen contiene tres oficinas secundarias. Una oficina secundaria puede contener un número cualquiera de elementos secundarios **Contact**. Los objetivos de la asignación son los siguientes:

- Extraer todos los elementos **Contact** del archivo XML de origen y escribirlos en el archivo XML de destino.
- Asignar a cada contacto un número de identificación único (el elemento **ID** del archivo XML de destino).
- El ID de cada contacto debe tener el formato **CXX-YYYYY**, donde X identifica el número de oficina e Y el número de contacto. Si el número de oficina ocupa menos que dos caracteres es necesario añadirle ceros a la izquierda. Asimismo, si el número de contactos ocupa menos que cinco caracteres entera ocupe necesario añadirle ceros a la izquierda. En consecuencia, el número identificativo del primer contacto de la primera oficina sería **C01-00001**.

Para conseguir los objetivos de la asignación se han usado varias funciones de MapForce, entre las que está la función **position**. La función **position** obtiene la posición de cada oficina. La siguiente obtiene la posición de cada uno de los contactos dentro de cada oficina.

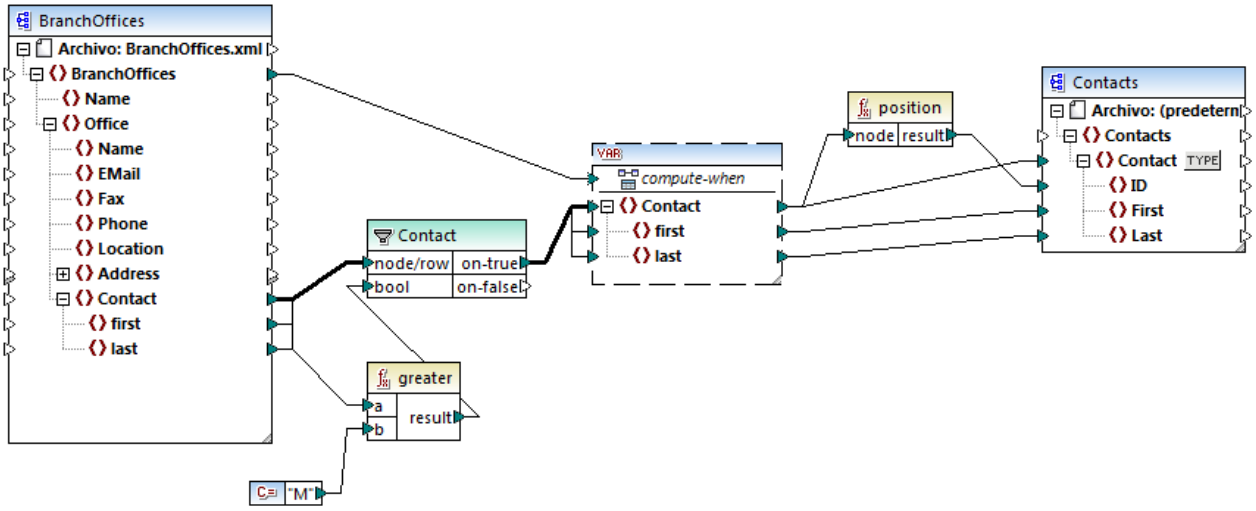
Al usar la función **position** es importante considerar el contexto de asignación actual 409. Concretamente, al ejecutar la asignación se establece el contexto inicial desde el elemento raíz del componente de destino al elemento de origen al que está conectado (aunque sea de forma indirecta mediante funciones). En este ejemplo, la función **position** de más arriba procesa la secuencia de todas las oficinas y genera inicialmente el valor 1, que corresponde a la primera oficina de la secuencia. La función **position** de más abajo genera números secuenciales que corresponden a la posición del contacto en el contexto de esa oficina (1, 2, 3, etc.). Tenga en cuenta que esta secuencia "interna" se reinicia y vuelve a empezar desde el 1 cuando se procesa la oficina siguiente. Las dos funciones **pad-string-left** usan ceros de relleno en los números generados, tal y como se explica más arriba. La función `concat` opera en el contexto de cada contacto (debido a la conexión principal del componente de entrada al **Contact** de destino). La función junta todos los valores computados y devuelve el número de identificación único de cada contacto.

A continuación mostramos el resultado que genera la asignación anterior (tenga en cuenta que hemos eliminado algunos de los registros para que ek código se pueda leer mejor):

```
<Contacts>
 <Contact>
 <ID>C01-00001</ID>
```

```
<First>Vernon</First>
<Last>Callaby</Last>
</Contact>
<Contact>
<ID>C01-00002</ID>
<First>Frank</First>
<Last>Further</Last>
</Contact>
<!-- ... -->
<Contact>
<ID>C02-00001</ID>
<First>Steve</First>
<Last>Meier</Last>
</Contact>
<Contact>
<ID>C02-00002</ID>
<First>Theo</First>
<Last>Bone</Last>
</Contact>
<!-- ... -->
</Contacts>
```

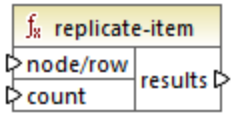
También puede haber casos en los que tenga que obtener la posición de los elementos resultantes una vez aplicado un filtro[178]. Tenga en cuenta que el componente filtro no es una función de secuencia, por lo que no se puede usar directamente junto con la función **position** para buscar la posición de elementos filtrados. Esto se puede hacer de forma indirecta añadiendo un componente[160] de variable a la asignación. Por ejemplo, la asignación siguiente es una versión simplificada de la anterior. El archivo de diseño de esta asignación está en: **<Documentos>\Altova\MapForce2024\MapForceExamples\PositionInFilteredSequence.mfd**.



Los resultados de los componentes de variable siempre son secuencias. Por tanto, en la asignación anterior la función **position** recorre la secuencia creada por la variable y devuelve la posición de cada uno de los elementos de esa secuencia. Esta asignación se analiza más detalladamente en Ejemplo: filtrar y numerar nodos[168].

6.6.9.15 replicate-item

Repite cada elemento de la secuencia de entrada tantas veces como se indique el argumento **count** Si conecta un único elemento a la secuencia **node/row** de entrada, la función devuelve elementos *N* donde *N* es el valor del argumento **count** Si conecta una secuencia de elementos a la secuencia de entrada **node/row**, la función repite cada elemento individual de la secuencia tantas veces como indique **count** y los procesa uno a uno. Por ejemplo, si count es **2**, entonces la secuencia `1,2,3` produce `1,1,2,2,3,3`. Repite cada elemento de la secuencia de entrada tantas veces como se indique en el argumento **count**.



Lenguajes

Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

Parámetros

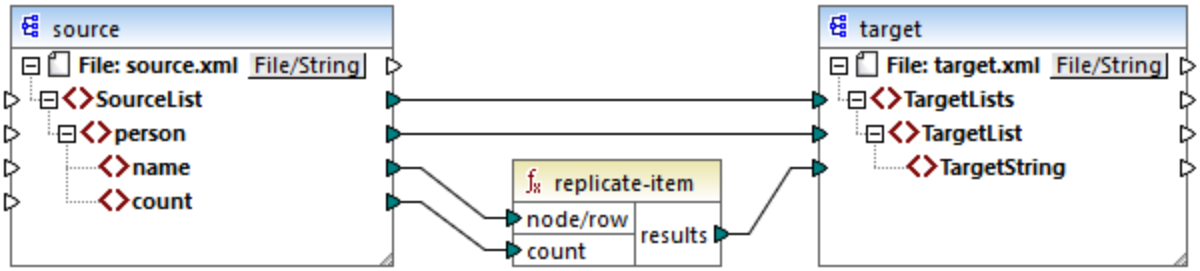
Nombre	Descripción
nodo/fila	Esta entrada debe recibir una conexión desde un elemento de la asignación que suministre una secuencia 408 de cero o más valores. Por ejemplo, la conexión puede provenir de un elemento XML de origen.
count	Indica el número de veces que se debe replicar cada elemento o secuencia que estén conectados a **node/row**.

Ejemplo

Por ejemplo, imagine que tiene un archivo XML de origen con esta estructura:

```
<SourceList>
 <person>
 <name>Michelle</name>
 <count>2</count>
 </person>
 <person>
 <name>Ted</name>
 <count>4</count>
 </person>
 <person>
 <name>Ann</name>
 <count>3</count>
 </person>
</SourceList>
```

Con ayuda de la función **replicate-item** puede repetir cada nombre de persona las veces que quiera en el componente de destino. Para conseguirlo conecte el nodo <count> de cada persona a la entrada count de la función **replicate-item**:

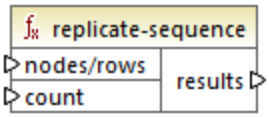


Este sería el resultado:

```
<TargetLists>
 <TargetList>
 <TargetString>Michelle</TargetString>
 <TargetString>Michelle</TargetString>
 </TargetList>
 <TargetList>
 <TargetString>Ted</TargetString>
 <TargetString>Ted</TargetString>
 <TargetString>Ted</TargetString>
 <TargetString>Ted</TargetString>
 </TargetList>
 <TargetList>
 <TargetString>Ann</TargetString>
 <TargetString>Ann</TargetString>
 <TargetString>Ann</TargetString>
 </TargetList>
</TargetLists>
```

6.6.9.16 replicate-sequence

Repite todos los elementos de la secuencia de entrada tantas veces como indique el argumento **count**. Por ejemplo, si count es **2**, entonces la secuencia (**1,2,3**) produce (**1,2,3,1,2,3**).



Lenguajes

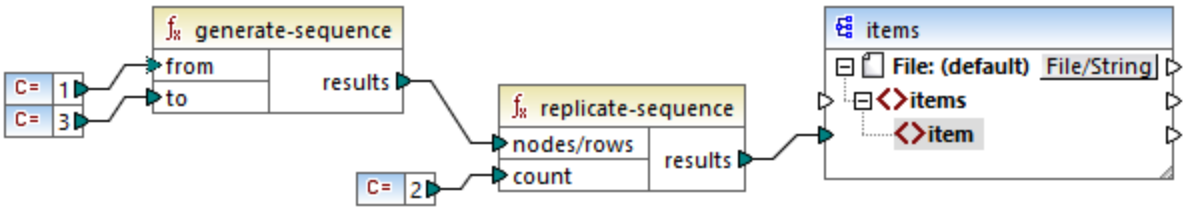
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

Parámetros

Nombre	Descripción
nodos/filas	Esta entrada debe recibir una conexión desde un elemento de la asignación que suministre una secuencia[408] de cero o más valores. Por ejemplo, la conexión puede provenir de un elemento XML de origen.
count	Indica el número de veces que se debe replicar la secuencia vinculada.

Ejemplo

La asignación de ejemplo siguiente genera la secuencia `1,2,3`. Esta secuencia es procesada por la función `replicate-sequence`, que escribe el resultado en un archivo XML de destino.

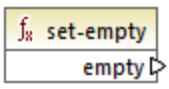


El argumento **count** tiene el valor **2**, por lo que la secuencia se replica dos veces y después pasa al archivo de destino. En consecuencia, la salida de la asignación es la siguiente (sin incluir las declaraciones de esquema XML):

```
<items>
 <item>1</item>
 <item>2</item>
 <item>3</item>
 <item>1</item>
 <item>2</item>
 <item>3</item>
</items>
```

6.6.9.17 set-empty

Devuelve una secuencia vacía.

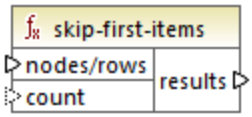


Lenguajes

Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

6.6.9.18 skip-first-items

Pasa por alto los X primeros nodos/elementos de la secuencia de entrada (siendo X el número dado por el parámetro **count**) y devuelve el resto de la secuencia.



Lenguajes

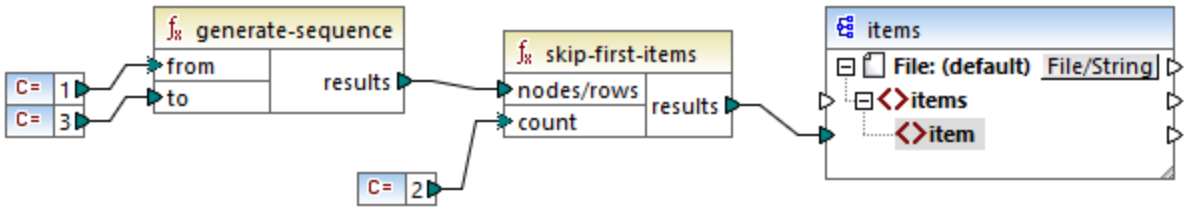
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

Parámetros

Nombre	Descripción
nodos/filas	Esta entrada debe recibir una conexión desde un elemento de la asignación que suministre una secuencia[408] de cero o más valores. Por ejemplo, la conexión puede provenir de un elemento XML de origen.
count	Argumento opcional. Indica el número de elementos a skip. El valor predeterminado es **1**.

Ejemplo

La asignación de ejemplo siguiente genera la secuencia `1,2,3`. Esta secuencia es procesada por la función `skip-first-items`, que escribe el resultado en un archivo XML de destino.

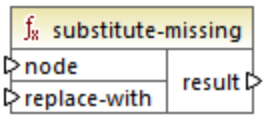


El argumento **count** tiene el valor **2**, por lo que los primeros dos elementos se omiten y el resto se pasan al archivo de destino. En consecuencia, la salida de la asignación es la siguiente (sin incluir las declaraciones de esquema XML):

```
<items>
 <item>3</item>
</items>
```

6.6.9.19 substitute-missing

Esta función es una combinación de la función exists[281] y la condición if-else.[181]. Si el elemento que está conectado a la entrada del **nodo** existe, se copia su contenido en el archivo de destino. De lo contrario utiliza el elemento asignado al parámetro **replace-with**.



Lenguajes

Built-in, C++, C#, Java, XQuery#, XSLT 1.0 y, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Parámetros

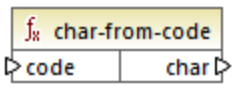
Nombre	Descripción
nodo	Esta entrada debe recibir una conexión desde un elemento de la asignación que suministre una secuencia[408] de cero o más valores. Por ejemplo, la conexión puede provenir de un elemento XML de origen.
replace-with	Este elemento de entrada estar conectado a un elemento de la asignación que proporcione el valor de reemplazo.

6.6.10 core | string functions (cadena)

La biblioteca core | string ofrece las funciones de cadena más comunes para manipular diferentes tipos de datos de origen y extraer porciones, comprobar si existen subcadenas o recuperar información de las cadenas.

6.6.10.1 char-from-code

El resultado es el carácter que represente el valor (código) Unicode decimal del parámetro value. **Consejo:** Para encontrar el valor Unicode decimal de un carácter puede usar la función `code-from-char`[309] .



Lenguajes

Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

Parámetros

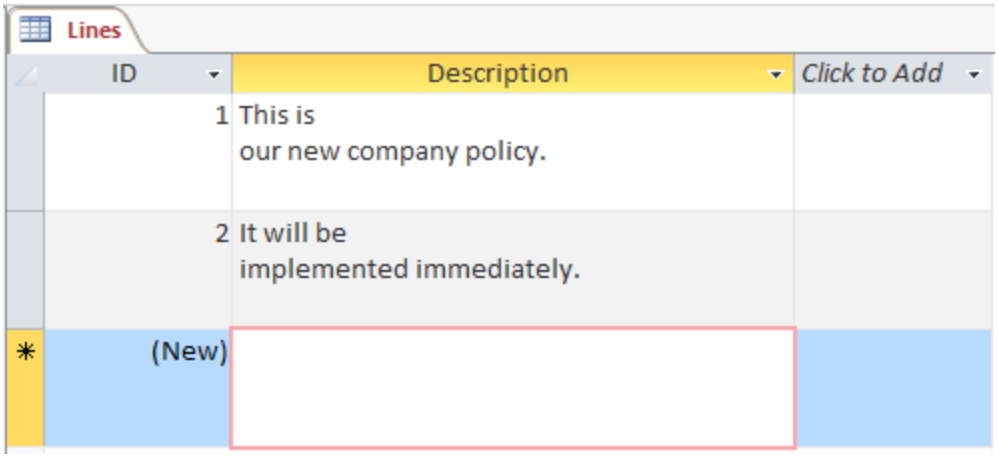
Nombre	Descripción
código	El valor Unicode como número decimal.

Ejemplo 1

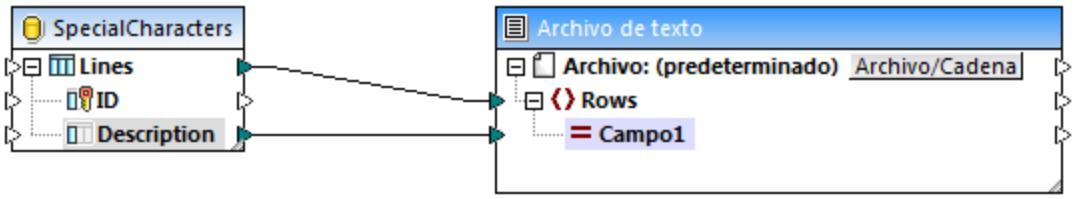
De acuerdo con los gráficos de la página web de Unicode (https://www.unicode.org/charts/), el carácter de exclamación tiene el valor hexadecimal `0021`. El valor correspondiente en formato decimal es **33**. Por tanto, si el argumento de la función `char-from-code` es `33`, el resultado será el carácter `!`.

Ejemplo 2 (ediciones Professional y Enterprise)

En este ejemplo se puede ver cómo reemplazar caracteres especiales en una base de datos con caracteres de espacio. Imagine que tiene una base de datos SQLite compuesta por una tabla llamada "Lines" que tiene dos columnas: "ID" y "Description".



El objetivo es extraer cada descripción a un archivo CSV (donde hay una descripción por línea). La asignación de datos que necesitamos para conseguir este objetivo es:

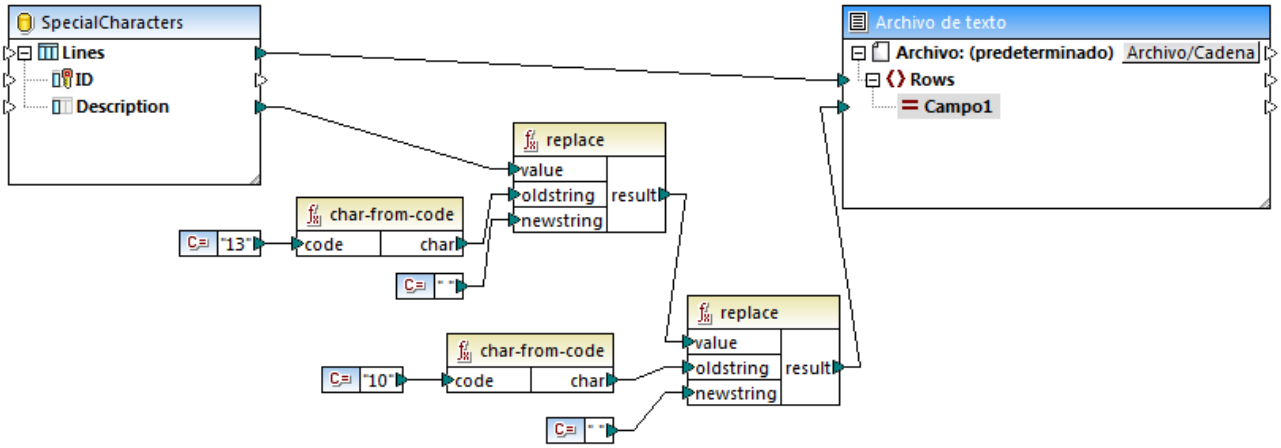


Sin embargo, como cada fila "Description" de Access contiene varias líneas separadas por caracteres CR/LF, el resultado de la asignación incluye también saltos de línea y este no es el resultado que queremos.

```
"This is
our new company policy."
"It will be
implemented immediately."
```

Para resolver este problema añadiremos las funciones **char-from-code** y **replace** de la biblioteca de MapForce. Cada descripción debe procesarse de modo que, cuando se detecten caracteres especiales, éstos se reemplazarán con un espacio.

En el gráfico Unicode (http://www.unicode.org/charts/), los caracteres LF y CR corresponden a los caracteres **hex 0A | dec 10** y **hex 0D | dec 13** respectivamente. Por tanto, la asignación debe modificarse para convertir los valores Unicode decimales 13 y 10 en una cadena para que la función **replace** los pueda procesar.

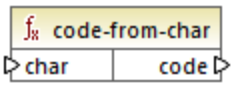


En la vista previa de resultados de esta asignación podemos ver que los caracteres CR/LF de cada campo de la base de datos se reemplazaron con un espacio.

```
This is  our new company policy.
It will be  implemented immediately.
```

6.6.10.2 code-from-char

El resultado es el valor Unicode decimal del carácter dado como argumento. Si la cadena dada como argumento tiene varios caracteres, entonces el resultado es el valor Unicode del primer carácter.



Lenguajes

Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

Parámetros

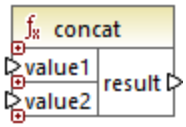
Nombre	Descripción
char	El valor string de entrada.

Ejemplo

Si el parámetro de entrada **char** es el carácter **$** (signo del dólar), la función devuelve **36** (que es el valor Unicode decimal de este carácrer).

6.6.10.3 concat

Concatena (anexa) dos o más valores dando lugar a una sola cadena. Todos los valores de entrada se convierten automáticamente en valores de tipo "string". Por defecto, esta función tiene sólo dos parámetros, pero puede añadir más. Haga clic en el icono **Agregar parámetro** (⊡) o **Eliminar parámetro** (⊠ para añadir o borrar parámetros.



Nota: Todos los componentes de entrada de la función `concat` deben tener un valor. Si falta el valor en alguno de ellos no se llama a la función y se genera un error. Tenga en cuenta que una cadena vacía es un valor de entrada válido; sin embargo, una secuencia vacía (como el resultado de la función **`set-empty`**) no es un valor válido, por lo que en ese caso la función fallará. Para evitar eso puede procesar primero los valores con la función substitute-missing 307 y después usar el resultado como componente de entrada de la función **`concat`**.

Lenguajes

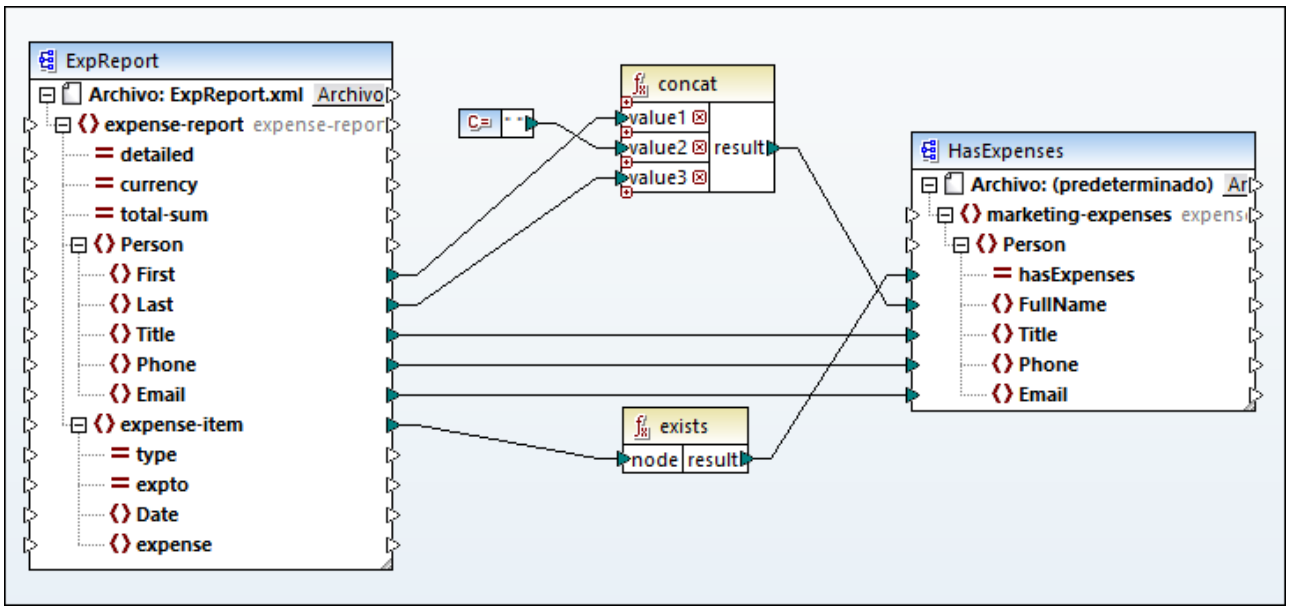
Built-in, C++, C#, Java, XQuery#, XSLT 1.0 y, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Parámetros

Nombre	Descripción
value1	El primer valor de la entrada.
value2	El segundo valor de la entrada.
valueN	El valor de *n* entrada.

Ejemplo

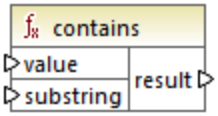
En la asignación siguiente, la función **concat** junta el nombre, la constante "" y el apellido. El valor de retorno se escribe en el elemento de destino **FullName**. La asignación de esta función se encuentra en: **<Documentos>\Altova\MapForce2024\MapForceExamples\HasMarketingExpenses.mfd**.



HasMarketingExpenses.mfd

6.6.10.4 contains

El resultado es **true** si la cadena de entrada (string) empieza por la subcadena (susbtr).



Lenguajes

Built-in, C++, C#, Java, XQuery#, XSLT 1.0 y, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Parámetros

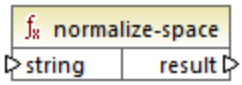
Nombre	Descripción
value	El valor de entrada (el "pajar").
substring	La subcadena que se busca (la "aguja").

Ejemplo

El resultado es true si la entrada **value** es "category" y **substring** es "cat".

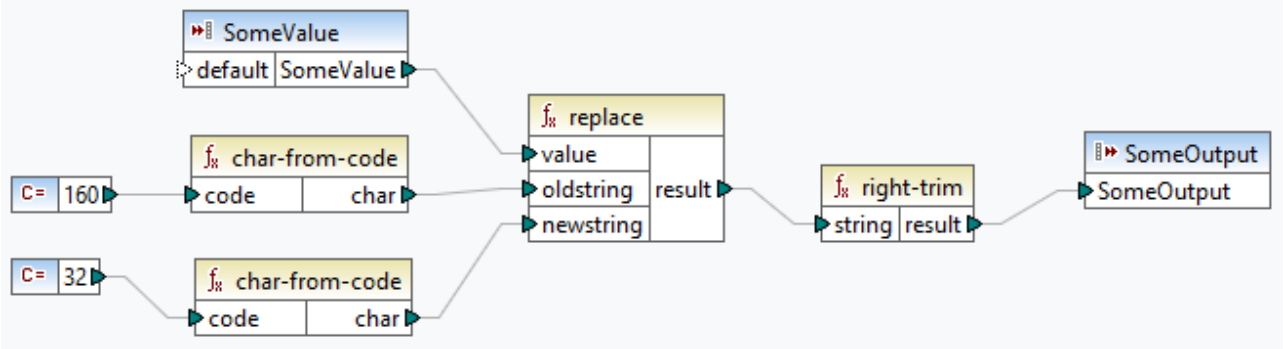
6.6.10.5 normalize-space

El resultado es la cadena de entrada normalizada. Es decir, se eliminan los espacios iniciales y finales y cada secuencia de espacios en blancos consecutivos se reemplaza con un solo espacio en blanco. El carácter Unicode para el espacio es (U+0020).



Acerca de los espacios duros

Las funcione **`left-trim`**, **`right-trim`** y **`normalize-space`** no eliminan los espacios duros. Una posible solución es reemplazar esos caracteres, cuya representación decimal es 160, por el carácter espacio, cuya representación decimal es 32. En la asignación siguiente puede ver que, una vez se ha reemplazado el espacio duro, el valor `SomeValue` reducido se asigna al destino.



Si su componente de origen es un archivo de Excel, puede eliminar espacios extra en Excel combinando las funciones TRIM, CLEAN y SUBSTITUTE. Para más detalles consulte *Las diez formas principales de limpiar los datos*.

Lenguajes

Built-in, C++, C#, Java, XQuery, XSLT 1.0 y, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Parámetros

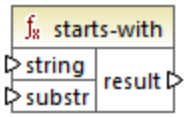
Nombre	Descripción
string	La cadena de entrada que se quiere normalizar.

Ejemplo

Si la cadena de entrada es **`El veloz zorro marrón`**, la función devuelve **`El veloz zorro marrón`**.

6.6.10.6 starts-with

El resultado es **true** si la cadena de entrada (string) empieza por la subcadena (susbtr). De lo contrario, el resultado es **false**



Lenguajes

Built-in, C++, C#, Java, XQuery#, XSLT 1.0 y, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Parámetros

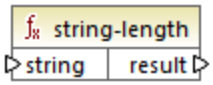
Nombre	Descripción
string	La cadena de entrada.
substr	La subcadena que se debe comprobar.

Ejemplo

Si la cadena de entrada **string** es `category` y la subcadena es `cat`, la función devuelve **true**.

6.6.10.7 string-length

El resultado es el número de caracteres asignados al parámetro de entrada string.



Lenguajes

Built-in, C++, C#, Java, XQuery#, XSLT 1.0 y, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Parámetros

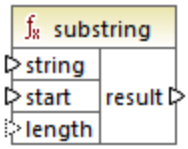
Nombre	Descripción
string	La cadena de entrada.

Ejemplo

Si la cadena de entrada es `car`, la función devuelve **3**. Si la cadena de entrada es **una cadena vacía** , la función devuelve **0**.

6.6.10.8 substring

El resultado es la subcadena (fragmento de cadena) del parámetro de entrada string, siendo **start** la posición del carácter de inicio y **length** la longitud de la subcadena.



Lenguajes

Built-in, C++, C#, Java, XQuery#, XSLT 1.0 y, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Parámetros

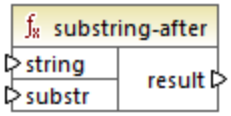
Nombre	Descripción
string	La cadena de entrada.
start	Indica la posición inicial (índice) a partir de la cual se debe obtener la subcadena. El primer índice es **1**.
length	Opcional. Indica el número de caracteres que se obtienen. Si no se indica el parámetro length, el resultado es un fragmento que empieza en la posición de inicio y termina en la posición final de la cadena.

Ejemplo

P. ej., si la cadena de entrada es `MapForce` start es **1** y **length** es **3**, la función devuelve `Map`. P. ej., si la cadena de entrada es `MapForce` start es **4** y no se indica **length**, la función devuelve `Force`.

6.6.10.9 substring-after

El resultado es el fragmento de la cadena string a partir de la primera aparición del parámetro **substr**. Si la cadena del parámetro **substr** no aparece en el parámetro **string**, el resultado de la función es una cadena vacía.



Lenguajes

Built-in, C++, C#, Java, XQuery#, XSLT 1.0 y, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Parámetros

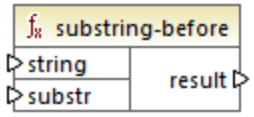
Nombre	Descripción
string	La cadena de entrada.
substr	La subcadena. El resultado de la función son los caracteres que haya después de la primera ocurrencia de **substr**.

Ejemplo

P. ej., si la cadena de entrada es `MapForce`, y **substr** es `Map`, la función devuelve `Force`. P. ej., si la cadena de entrada es `2020/01/04` y **substr** es `/` da como resultado la subcadena `01/04`.

6.6.10.10 substring-before

El resultado es el fragmento de la cadena del parámetro string hasta la primera aparición de los caracteres del parámetro **substr**. Si la cadena del parámetro **substr** no aparece en el parámetro **string**, el resultado de la función es una cadena vacía.



Lenguajes

Built-in, C++, C#, Java, XQuery#, XSLT 1.0 y, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Parámetros

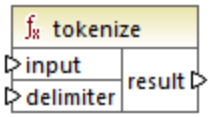
Nombre	Descripción
string	La cadena de entrada.
substr	La subcadena. Los caracteres que hay antes de la primera aparición de **substr** son el resultado de la función.

Ejemplo

P. ej., si la cadena de entrada es `MapForce`, y **substr** es `Force`, la función devuelve `Map`. Si la cadena de entrada es `2020/01/04` y **substr** es `/` da como resultado la subcadena `2020`.

6.6.10.11 tokenize

El resultado es la cadena input dividida en una secuencia formada por caracteres delimitados por el parámetro delimiter.



Lenguajes

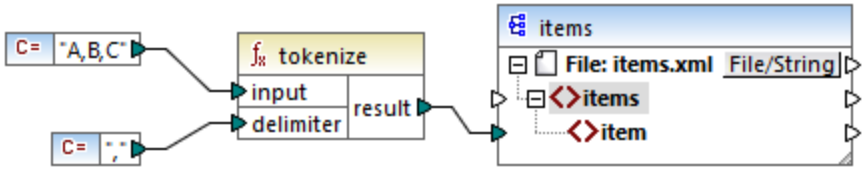
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

Parámetros

Nombre	Descripción
input	La cadena de entrada.
delimiter	El delimitador que se usa.

Ejemplo

P. ej. la cadena input es `A,B,C` y el delimitador es ",". El resultado será `A`, `B` y `C`.

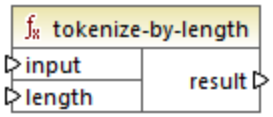


En la asignación de prueba de más arriba, el resultado de la función es una secuencia de cadenas. Según las reglas generales de asignación, 408 por cada elemento de la secuencia de origen se debe crear un **elemento** nuevo en el componente de destino. En consecuencia, el resultado de la asignación tiene este aspecto:

```
<items>
 <item>A</item>
 <item>B</item>
 <item>C</item>
</items>
```

6.6.10.12 tokenize-by-length

El resultado es la cadena de entrada dividida en una secuencia de cadenas, El parámetro **length** determina el tamaño de las cadenas resultantes.



Lenguajes

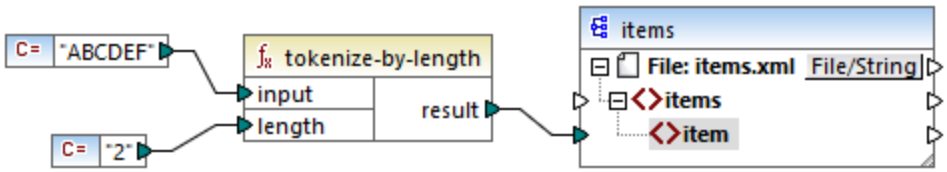
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

Parámetros

Nombre	Descripción
input	La cadena de entrada.
length	Determina la longitud de cada una de las cadenas que forman la secuencia de cadenas generada.

Ejemplo

P. ej., si la cadena input es **ABCDEF** y el parámetro length es **2**, entonces la función devuelve una secuencia de tres cadenas: **AB**, **CD** y **EF**.



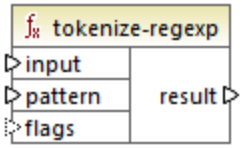
En la asignación de prueba de más arriba, el resultado de la función es una secuencia de cadenas. Según las reglas generales de asignación, 408 por cada elemento de la secuencia de origen se debe crear un **elemento** nuevo en el componente de destino.. En consecuencia, el resultado de la asignación tiene este aspecto:

```
<items>
 <item>AB</item>
 <item>CD</item>
 <item>EF</item>
</items>
```

6.6.10.13 tokenize-regexp

El resultado es la cadena de entrada dividida en una secuencia de cadenas, en la que la expresión regular **pattern** define el separador. El parámetro result no genera los separadores.

Nota: al generar código C++, C# o Java, las características avanzadas de la sintaxis de la expresión regular pueden variar ligeramente. Consulte la documentación regex de cada lenguaje para más información.



Lenguajes

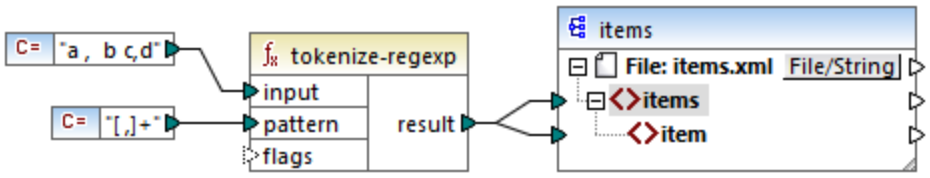
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

Parámetros

Nombre	Descripción
input	La cadena de entrada.
pattern	Indica una expresión regular pattern. Las cadenas que coincidan con esa expresión regular pattern se consideran delimitadores. Para más información consulte Expresiones regulares 231.
flags	Parámetro opcional. Indica expresiones regulares flags 233 . Por ejemplo, la expresión regular flag "i" indica al proceso de la asignación que opere sin tener en cuenta mayúsculas y minúsculas.

Ejemplo

El objetivo de la asignación siguiente es separar la cadena `a , b c,d` en una secuencia de cadenas en la que cada letra es un elemento de la secuencia. Los espacios o las comas redundantes deben eliminarse.



Para ello se pasa la expresión regular `[ ,]+` como parámetro a la función `tokenize-regexp`. Al usar esa expresión regular:

- La expresión regular pattern define una clase de caracteres `[ ,]`, de los cuales se usará un solo carácter (es decir, o bien el espacio, o bien la coma).

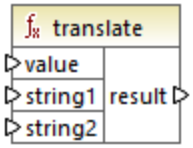
- El cuantificador **+** especifica "una o más" apariciones de la clase de caracteres/cadena. Sin este cuantificador, en la secuencia resultante se crearía un elemento por cada espacio o coma, que no es lo que pretendemos.

El resultado de la asignación sería:

```
<items>
 <item>a</item>
 <item>b</item>
 <item>c</item>
 <item>d</item>
</items>
```

6.6.10.14 translate

Reemplaza caracteres uno a uno. Los caracteres de la cadena **string1** (cadena de búsqueda) se reemplazan con los caracteres de la cadena **string2** (cadena de reemplazo) en la cadena de entrada **value**. Si en la cadena **string2** no hay caracteres que correspondan, entonces se elimina el carácter.



Lenguajes

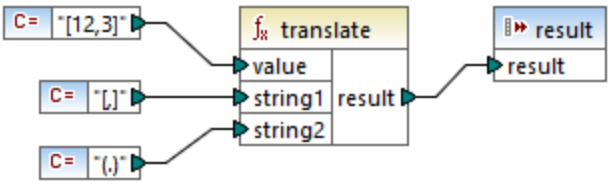
Built-in, C++, C#, Java, XQuery#, XSLT 1.0 y, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Parámetros

Nombre	Descripción
value	La cadena de entrada.
string1	Suministra una lista de caracteres de búsqueda. La posición de cada carácter en la cadena es importante.
string2	Suministra una lista de caracteres de reemplazo. La posición de cada uno de los caracteres de reemplazo debe tener su correspondencia en la cadena **string1**.

Ejemplo

Imagine que quiere convertir la cadena `[12,3]` en `(12.3)`. Es decir, quiere reemplazar los corchetes por paréntesis y la coma por un punto. Para obtener ese resultado puede usar la función `translate`:



En la asignación anterior, la primera constante pasa la cadena de entrada que se debe procesar. La segunda y la tercera constantes pasan una lista de caracteres como **string1** y **string2** respectivamente.

string1 `[,]`

string2 `(.)`

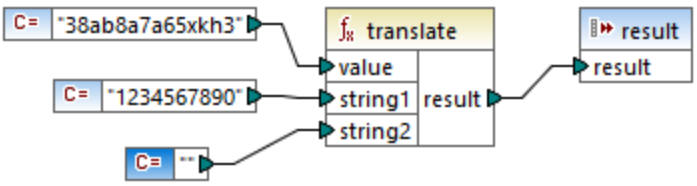
Tenga en cuenta que tanto **string1** como **string2** tienen el mismo número de caracteres. Cada carácter de **string1** se reemplaza con el carácter que esté en esa misma posición en **string2**. Por tanto, en este caso lo que ocurrirá es que:

- Cada carácter `[` será reemplazado por un carácter `(`
- Cada carácter `,` será reemplazado por un carácter `.`
- Cada carácter `]` será reemplazado por un carácter `)`

El resultado de la asignación sería:

```
(12.3)
```

Esta función también se puede usar para eliminar determinados caracteres de una cadena. Para ello, indique con el parámetro **string1** los caracteres que quiere eliminar y deje vacío el parámetro **string2**. Por ejemplo, esta asignación elimina tods los dígitos de la cadena **38ab8a7a65xkh3**.



El resultado de la asignación sería:

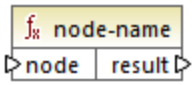
```
abaaxkh
```

6.6.11 xpath2 | accessors (descriptores de acceso)

Las funciones de la biblioteca **xpath2 | accessors** obtienen información sobre nodos o elementos XML. Estas funciones están disponibles cuando se seleccionan los lenguajes XSLT2 o XQuery.

6.6.11.1 base-uri

La función **base-uri** toma un argumento de nodo como entrada y devuelve el URI del recurso XML que contiene el nodo El salida es de tipo `xs: string`.



Lenguajes

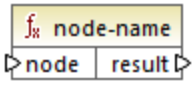
XQuery, XSLT 2.0, XSLT 3.0.

Parámetros

Nombre	Tipo	Descripción
nodo	**mf:node**	El nodo de entrada.

6.6.11.2 node-name

La función **node-name** toma un nodo como argumento de entrada y devuelve su QName. Cuando el Qname se representa como una cadena, toma la forma de `prefix:localname` si el nodo tiene un prefijo, o `localname` si no lo tiene. Para obtener el URI del espacio de nombres de un nodo, use la función **namespace-uri-from-QName** 279 .



Lenguajes

XQuery, XSLT 2.0, XSLT 3.0.

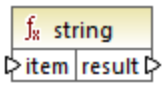
Parámetros

Nombre	Tipo	Descripción
nodo	**mf:node**	El nodo de entrada.

6.6.11.3 string

La función **string** funciona como el constructor `xs:string`: convierte su argumento en `xs:string`.

Cuando el argumento de entrada es un valor de un tipo atómico (por ejemplo `xs:decimal`), este valor atómico se convierte en un valor de tipo `xs:string`. Si el argumento de entrada es un nodo, se extrae el valor de cadena del nodo. (El valor de cadena de un nodo es una concatenación de los valores de los descendientes del nodo.)



Lenguajes

XQuery, XSLT 2.0, XSLT 3.0.

Parámetros

Nombre	Tipo	Descripción
item	`mf:item`	El valor de entrada.

6.6.12 xpath2 | anyURI functions

La biblioteca secundaria **xpath2 | anyURI** contiene la función **`resolve-uri`**. Esta función está disponible cuando se seleccionan los lenguajes XSLT2 o XQuery.

6.6.12.1 resolve-uri

La función **`resolve-uri`** toma un URI relativo como su primer argumento y lo resuelve a partir del URI de base del segundo argumento. El resultado es del tipo de datos `xs: string`. La implementación de la función trata las dos entradas como cadenas y no se comprueba si los recursos que identifican esos URIs existen realmente.

Lenguajes

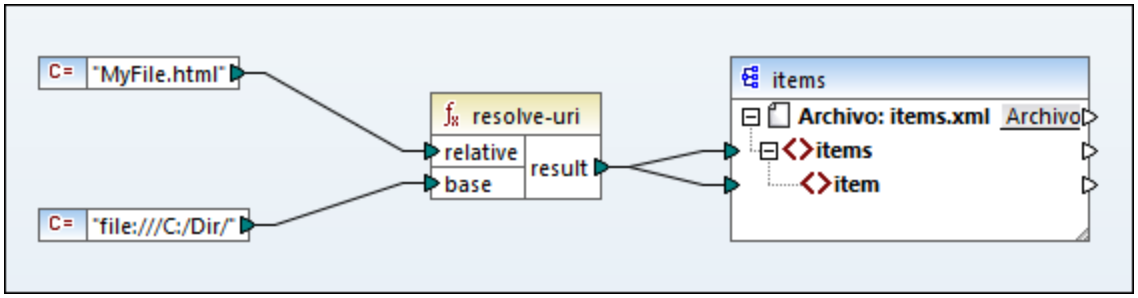
XQuery, XSLT 2.0, XSLT 3.0.

Parámetros

Nombre	Tipo	Descripción
relative	`xs:string`	El URI relativo que se quiere resolver a partir de la base.
base	`xs:string`	El URI de base.

Ejemplo

En la asignación siguiente, el primer argumento indica el URI relativo `MyFile.html` y el segundo el URI de base `file:///C:/Dir/`. El URI resuelto será una concatenación de ambos, es decir `file:///C:/Dir/MyFile.html`.

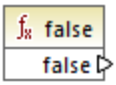


6.6.13 xpath2 | boolean functions (booleanas)

Las funciones booleanas de la biblioteca xpath2 | boolean son las funciones `true` y `false`. Estas funciones no toman ningún argumento y devuelven los valores booleanos de constante `true` y `false` respectivamente Estas funciones se pueden utilizar cuando se necesite un valor booleano de constante.

6.6.13.1 false

Devuelve el valor booleano **false**.

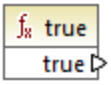


Lenguajes

XQuery, XSLT 2.0, XSLT 3.0.

6.6.13.2 true

Devuelve el valor booleano **true**.



Lenguajes

XQuery, XSLT 2.0, XSLT 3.0.

6.6.14 xpath2 | constructors (constructores)

Las funciones constructoras de la biblioteca de funciones xpath2 construyen ciertos tipos de datos a partir del texto de entrada. La tabla siguiente enumera las funciones constructoras disponibles.

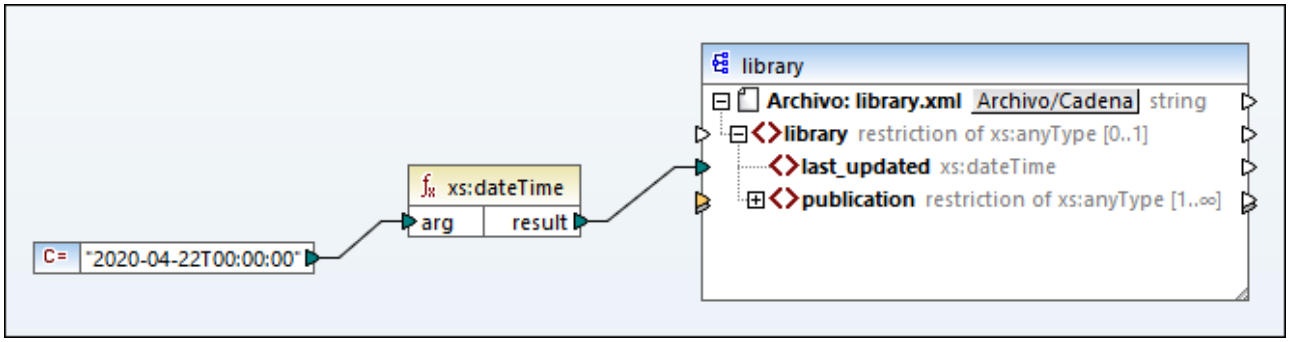
xs:ENTITY	**xs:double**	**xs:nonPositiveInteger**
xs:ID	**xs:duration**	**xs:normalizedString**
xs:IDREF	**xs:float**	**xs:positiveInteger**
xs:NCName	**xs:gDay**	**xs:short**
xs:NMTOKEN	**xs:gMonth**	**xs:string**
xs:Name	**xs:gMonthDay**	**xs:time**
xs:QName	**xs:gYear**	**xs:token**
xs:anyURI	**xs:gYearMonth**	**xs:unsignedByte**
xs:base64Binary	**xs:hexBinary**	**xs:unsignedInt**
xs:boolean	**xs:int**	**xs:unsignedLong**
xs:byte	**xs:integer**	**xs:unsignedShort**
xs:date	**xs:language**	**xs:untypedAtomic**
xs:dateTime	**xs:long**	**xs:yearMonthDuration**
xs:dayTimeDuration	**xs:negativeInteger**	
xs:decimal	**xs:nonNegativeInteger**	

Lenguajes

XQuery, XSLT 2.0, XSLT 3.0.

Ejemplo

Por lo general, el formato léxico del texto de entrada debe ser el formato esperado del tipo de datos que se debe construir. De lo contrario, la transformación no se puede realizar. Por ejemplo, si desea construir un tipo de datos `xs:dateTime`, utilice la función constructor **`xs:dateTime`**. El texto de entrada debe tener el formato léxico del tipo de datos `xs:dateTime`, es decir: **`AAAA-MM-DD:mm:ss`** (imagen siguiente).



En la imagen anterior, observe que se utilizó la constante **"2020-04-28T00:00:00"** como argumento de entrada de la función. La entrada también podría obtenerse de un elemento del documento de origen. La función **xs:dateTime** devuelve el texto de entrada **2020-04-28T00:00:00**, que es del tipo de datos `xs:dateTime`.

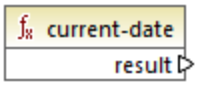
Al pasar el puntero del ratón encima del conector de entrada o de salida aparece el tipo de datos esperado para el elemento de la asignación (incluido el tipo de datos de los argumentos de la función).

6.6.15 xpath2 | context functions (contexto)

Las funciones de contexto de la biblioteca **xpath2 | context** ofrecen la hora y la fecha, la intercalación predeterminada utilizada por el procesador y el tamaño de la secuencia actual y la posición del nodo actual.

6.6.15.1 current-date

Devuelve la fecha actual (`xs:date`) del reloj del sistema.

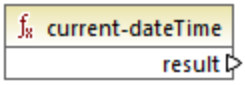


Lenguajes

XQuery, XSLT 2.0, XSLT 3.0.

6.6.15.2 current-dateTime

Devuelve la fecha y hora actual (`xs: dateTime`) del reloj del sistema.

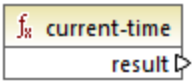


Lenguajes

XQuery, XSLT 2.0, XSLT 3.0.

6.6.15.3 current-time

Devuelve la hora actual (`xs:time`) del reloj del sistema.



Lenguajes

XQuery, XSLT 2.0, XSLT 3.0.

6.6.15.4 default-collation

La función **`default-collation`** no tiene argumentos y devuelve la intercalación predeterminada, es decir, la intercalación utilizada cuando no se especifica una intercalación para una función que la necesita.

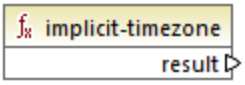
Las comparaciones (incluidas las funciones **`max-string`** y **`min-string`**) se basan en esta intercalación.

Lenguajes

XQuery, XSLT 2.0, XSLT 3.0.

6.6.15.5 implicit-timezone

Devuelve el valor de la propiedad "implicit timezone" del contexto de evaluación.

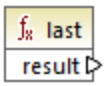


Lenguajes

XQuery, XSLT 2.0, XSLT 3.0.

6.6.15.6 last

Devuelve el número de elementos de la secuencia que se está procesando. Es importante recordar que la secuencia de los elementos la determina el contexto de asignación actual [409], como se describe en el ejemplo siguiente.



Lenguajes

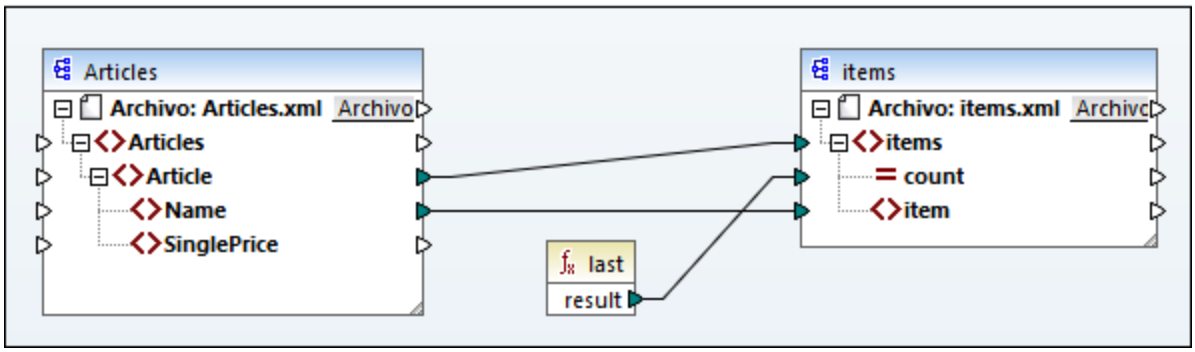
XQuery, XSLT 2.0, XSLT 3.0.

Ejemplo

Imagine que tiene que trabajar con este archivo XML de origen:

```
<Articles>
 <Article>
 <Name>T-Shirt</Name>
 <SinglePrice>25</SinglePrice>
 </Article>
 <Article>
 <Name>Socks</Name>
 <SinglePrice>2.30</SinglePrice>
 </Article>
 <Article>
 <Name>Jacket</Name>
 <SinglePrice>57.50</SinglePrice>
 </Article>
</Articles>
```

El objetivo es copiar datos en un archivo XML que tiene un esquema distinto. El recuento de todos los elementos también se debe guardar en el archivo XML de destino. Para ello puede usar una asignación como esta:



En el ejemplo anterior la función **last** devuelve la posición del último nodo del contexto primario actual y rellena el atributo **count** con el valor **3**.

```
<items count="3">
 <item>T-Shirt</item>
 <item>Pants</item>
 <item>Jacket</item>
</items>
```

Tenga en cuenta que el valor **3** es la posición del último elemento (y el recuento del total de elementos) en el contexto de la asignación creado por la conexión entre los elementos **Article** y **items**. Los elementos se copian en el archivo de destino aunque no exista esa conexión, pero en ese caso la función **last** devuelve el valor incorrecto **1** porque no tiene un contexto primario[415] que recorrer. (Más concretamente, usaría el contexto implícito predeterminado creado entre los elementos raíz de ambos componentes, que produce una secuencia de 1 elemento y no de 3, como se esperaba).

Se recomienda usar la función **count**[239] de la biblioteca **core** en lugar de la función **last** porque la primera tiene un argumento **parent-context** que permite alterar el contexto de la asignación de forma explícita.

6.6.16 xpath2 | durations, date, time functions (duración, fecha y hora)

Las funciones de duración, fecha y hora de la biblioteca **xpath2** permiten ajustar las fechas y horas al uso horario, extraer componentes de los datos fecha/hora y sustraer una unidad fecha/hora a otra.

Ajustar fechas y horas

Para ajustar los valores de hora y fecha de un uso horario siga estos pasos:

- **adjust-date-to-timezone**
- **adjust-date-to-timezone** (con argumento timezone)
- **adjust-dateTime-to-timezone**
- **adjust-dateTime-to-timezone** (con argumento timezone)
- **adjust-time-to-timezone**
- **adjust-time-to-timezone** (con argumento timezone)

Cada una de estas funciones toma un valor `date`, `time` o `dateTime` como primer argumento y ajusta los datos de entrada añadiendo, eliminando o modificando el componente de uso horario en función del valor del segundo argumento, si lo hay.

Si el primer argumento no contiene uso horario (por ejemplo, la fecha `2020-01` o la hora `14:00:00`) hay tres posibilidades.

- Si está presente el argumento **timezone**, el resultado contendrá el uso horario especificado en el segundo argumento. Se añade el uso horario en el segundo argumento.
- Si falta el argumento **timezone**, el resultado contendrá el uso horario implícito, es decir, el del sistema. Se añade el uso horario del sistema.
- Si el argumento **timezone** está vacío, está vacío, el resultado no contendrá el uso horario.

Si el primer argumento contiene un uso horario (por ejemplo, la fecha `2020-01-01+01:00` o la hora `14:00:00+01:00`) hay tres posibilidades.

- Si está presente el argumento **timezone**, el resultado contendrá el uso horario especificado en el segundo argumento. El uso horario original es reemplazado por el del segundo argumento.
- Si falta el argumento **timezone**, el resultado contendrá el uso horario implícito, es decir, el del sistema. El uso horario original es reemplazado por el del sistema.
- Si el argumento **timezone**, está vacío, el resultado no contendrá el uso horario.

Extraer componentes de fechas y horas

Estas son las funciones para extraer valores numéricos como horas, minutos, días, meses, etc. de valores de fecha y hora:

- **day-from-date**
- **day-from-dateTime**
- **hours-from-dateTime**
- **hours-from-time**
- **minutes-from-dateTime**
- **minutes-from-time**
- **month-from-date**
- **month-from-dateTime**
- **seconds-from-dateTime**
- **seconds-from-time**
- **timezone-from-date**
- **timezone-from-dateTime**
- **imezone-from-time**
- **year-from-date**
- **year-from-dateTime**

Cada una de estas funciones extrae un componente en concreto de los valores `xs:date`, `xs:time`, `xs:dateTime` y `xs:duration`. El resultado puede ser `xs:integer` o `xs:decimal`.

Funciones From

Estas son las funciones para extraer componentes de hora de duraciones:

- **days-from-duration**
- **hours-from-duration**
- **minutes-from-duration**
- **months-from-duration**
- **seconds-from-duration**
- **years-from-duration**

La duración se puede indicar como `xs:yearMonthDuration` (para extraer años o meses) o como `xs:dayTimeDuration` (para extraer días, horas, minutos y segundos). Todas las funciones devuelven un resultado de tipo `xs:integer` excepto la función **seconds-from-duration**, que devuelve .`xs:decimal`.

Funciones Subtract

Estas son las funciones para sustraer valores de fecha y hora:

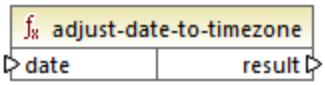
- **subtract-dateTimes**

- **subtract-dates**
- **subtract-times**

Cada una de estas funciones permite sustraer un valor de hora a otro y obtener un valor de duración como resultado.

6.6.16.1 adjust-date-to-timezone

Ajusta un valor `xs:date` a la zona horaria implícita en el contexto de evaluación (el huso horario del sistema).



Lenguajes

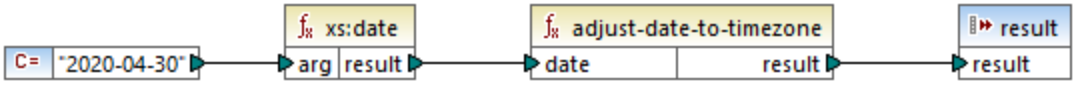
XQuery, XSLT 2.0.

Parámetros

Nombre	Tipo	Descripción
date	**xs:date**	El valor de entrada de tipo `xs:date`.

Ejemplo

Esta asignación genera un `xs:date` a partir de una cadena y lo pasa como argumento a la función **adjust-date-to-timezone**.

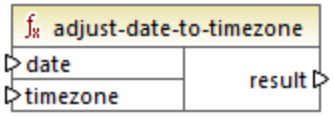


Asignación XSLT 2.0

Si la asignación se ejecuta en un equipo en el que el huso horario es +02:00, la función ajustará el valor de la fecha a este huso. En consecuencia, el resultado de la asignación será **2020-04-30+02:00**.

6.6.16.2 adjust-date-to-timezone

Ajusta un valor `xs:date` a un huso horario específico o a ninguno. Si el argumento **timezone** es una secuencia vacía, la función devuelve un elemento `xs:date` sin huso horario. De lo contrario lo devuelve con huso horario.



Lenguajes

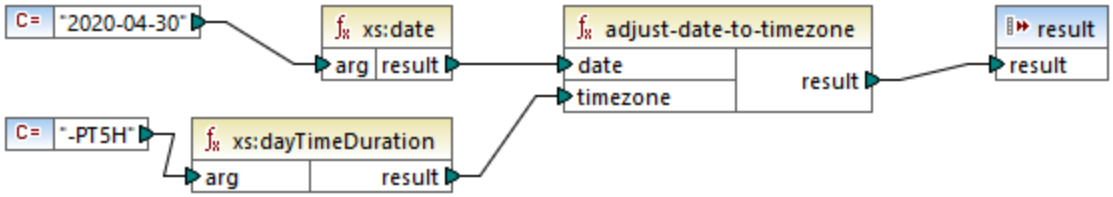
XQuery, XSLT 2.0.

Parámetros

Nombre	Tipo	Descripción
date	`xs:date`	El valor de entrada de tipo `xs:date`.
timezone	`xs:dayTimeDuration`	El huso horario expresado como un valor `xs:dayTimeDuration`. El valor puede ser negativo. Por ejemplo, un huso horario de -5 horas se puede expresar como `-PT5H`.

Ejemplo

Esta asignación genera los dos parámetros de la función **`adjust-date-to-timezone`** a partir de cadenas usando las funciones XPath 2 constructor[324] correspondientes. El objetivo de la asignación es ajustar el huso horario a -5 horas. Este huso horario se puede expresar como **`-PT5H`**.

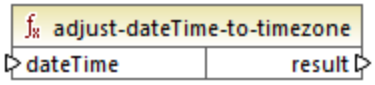


Asignación XSLT 2.0

La función ajusta el valor de la fecha al huso horario dado como argumento. En consecuencia, el resultado de la asignación es **`2020-04-30-05:00`**.

6.6.16.3 adjust-dateTime-to-timezone

Ajusta un valor `xs:dateTime` al huso horario implícito en el contexto de evaluación (el huso horario del sistema).



Lenguajes

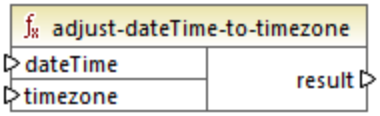
XQuery, XSLT 2.0.

Parámetros

Nombre	Tipo	Descripción
dateTime	`xs:dateTime`	El valor de entrada de tipo `xs:dateTime`.

6.6.16.4 adjust-dateTime-to-timezone

Ajusta un valor `xs:dateTime` a un huso horario específico o a ninguno. Si el argumento **timezone** es una secuencia vacía, la función devuelve un elemento `xs:dateTime` sin huso horario. De lo contrario lo devuelve con huso horario.



Lenguajes

XQuery, XSLT 2.0.

Parámetros

Nombre	Tipo	Descripción
dateTime	`xs:dateTime`	El valor de entrada de tipo `xs:dateTime`.
timezone	`xs:dayTimeDuration`	El huso horario expresado como un valor `xs:dayTimeDuration`. El valor puede ser negativo. Por ejemplo, un huso horario de -5 horas se puede expresar como `-PT5H`.

6.6.16.5 adjust-time-to-timezone

Ajusta un valor `xs:time` al huso horario implícito en el contexto de evaluación (el huso horario del sistema).

Lenguajes

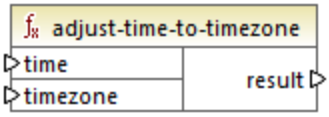
XQuery, XSLT 2.0.

Parámetros

Nombre	Tipo	Descripción
time	`xs:time`	El valor de entrada de tipo `xs:time`.

6.6.16.6 adjust-time-to-timezone

Ajusta un valor `xs:time` a un huso horario específico o a ninguno. Si el argumento **timezone** es una secuencia vacía, la función devuelve un elemento `xs:time` sin huso horario. De lo contrario lo devuelve con huso horario.



Lenguajes

XQuery, XSLT 2.0.

Parámetros

Nombre	Tipo	Descripción
time	`xs:time`	El valor de entrada de tipo `xs:time`.
timezone	`xs:dayTimeDuration`	El huso horario expresado como un valor `xs:dayTimeDuration`. El valor puede ser negativo. Por ejemplo, un huso horario de -5

Nombre	Tipo	Descripción
		horas se puede expresar como `-PT5H`.

6.6.16.7 day-from-date

Devuelve un elemento `xs:integer` que representa la parte del día del valor `xs:date` dado como argumento.

Lenguajes

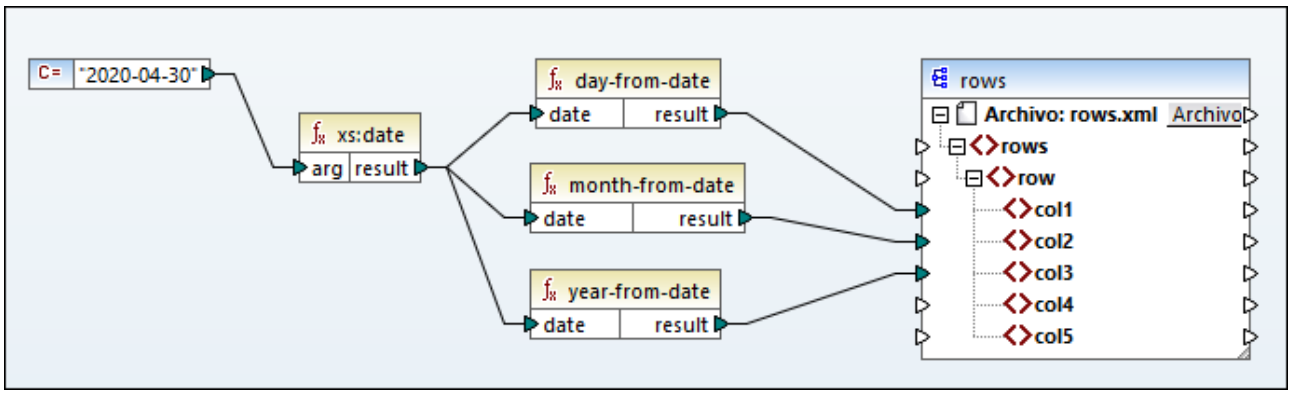
XQuery, XSLT 2.0.

Parámetros

Nombre	Tipo	Descripción
date	`xs:date`	El valor de entrada de tipo `xs:date`.

Ejemplo

Esta asignación convierte una cadena en `xs:date` con la función constructora **`xs:date`**. Las funciones **`day-from-date`**, **`month-from-date`** y **`year-from-date`** extraen las respectivas partes de la fecha y las escriben en un elemento aparte en el archivo XML de destino.



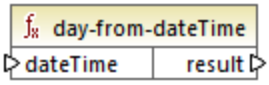
Asignación XQuery 1.0

El resultado de la asignación es:

```
<rows>
   <row>
      <col1>30</col1>
      <col2>4</col2>
      <col3>2020</col3>
   </row>
</rows>
```

6.6.16.8 day-from-dateTime

Devuelve un elemento `xs:integer` que representa la parte del día del valor `xs:dateTime` dado como argumento.



Lenguajes

XQuery, XSLT 2.0.

Parámetros

Nombre	Tipo	Descripción
dateTime	`xs:dateTime`	El valor de entrada de tipo `xs:dateTime`.

6.6.16.9 days-from-duration

Devuelve un elemento `xs:integer` que representa el componente "días" de la representación canónica del valor de duración dado como argumento.

Lenguajes

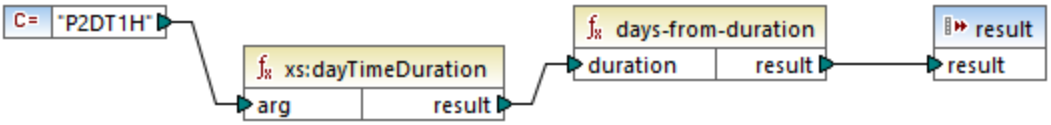
XQuery, XSLT 2.0.

Parámetros

Nombre	Tipo	Descripción
duration	`xs:duration`	El valor de entrada de tipo `xs:duration`.

Ejemplo

Esta asignación genera el `xs:dayTimeDuration` de **`P2DT1H`** (2 días y 1 hora) y lo para como entrada a la función **`days-from-duration`**. El resultado es **2**.



Asignación XSLT 2.0

Nota: si la duración es **`P1DT24H`** (2 días y 1 hora), la función devuelve **2**, no **1** porque la representación canónica de **`P1DT24H`** es **`P2D`** (2 días).

6.6.16.10 hours-from-dateTime

Devuelve un elemento `xs:integer` que representa la parte de las horas del valor `xs:dateTime` dado como argumento.

Lenguajes

XQuery, XSLT 2.0.

Parámetros

Nombre	Tipo	Descripción
dateTime	**`xs:dateTime`**	El valor de entrada de tipo `xs:dateTime`.

6.6.16.11 hours-from-duration

Devuelve un elemento `xs:integer` que representa el componente horas de la la representación canónica del valor de duración dado como argumento.

Lenguajes

XQuery, XSLT 2.0.

Parámetros

Nombre	Tipo	Descripción
duration	**`xs:duration`**	El valor de entrada de tipo `xs:duration`.

Ejemplo

Si la duración es `PT1H60M` (1 hora y 60 minutos), la función devuelve **2**, no **1** porque la representación canónica `PT1H60M` es `PT2H` (2 días).

6.6.16.12 hours-from-time

Devuelve un elemento `xs:integer` que representa la parte de las horas del valor `xs:time` dado como argumento.

Lenguajes

XQuery, XSLT 2.0.

Parámetros

Nombre	Tipo	Descripción
time	`xs:time`	El valor de entrada de tipo `xs:time`.

6.6.16.13 minutes-from-dateTime

Devuelve un elemento `xs:integer` que representa la parte de los minutos del valor `xs:dateTime` dado como argumento.

Lenguajes

XQuery, XSLT 2.0.

Parámetros

Nombre	Tipo	Descripción
dateTime	`xs:dateTime`	El valor de entrada de tipo `xs:dateTime`.

6.6.16.14 minutes-from-duration

Devuelve un elemento `xs:integer` que representa el componente "minutos" de la la representación canónica del valor de duración dado como argumento.

Lenguajes

XQuery, XSLT 2.0.

Parámetros

Nombre	Tipo	Descripción
duration	`xs:duration`	El valor de entrada de tipo `xs:duration`.

Ejemplo

Si la duración es `PT1M60S` (1 minuto y 60 segundos), la función devuelve **2**, no **1** porque la representación canónica `PT1M60S` es `PT2M` (2 minutos).

6.6.16.15 minutes-from-time

Devuelve un elemento `xs:integer` que representa la parte de los minutos del valor `xs:time` dado como argumento.

Lenguajes

XQuery, XSLT 2.0.

Parámetros

Nombre	Tipo	Descripción
time	`xs:time`	El valor de entrada de tipo `xs:time`.

6.6.16.16 month-from-date

Devuelve un elemento `xs:integer` que representa la parte del mes del valor `xs:date` dado como argumento.

Lenguajes

XQuery, XSLT 2.0.

Parámetros

Nombre	Tipo	Descripción
date	`xs:date`	El valor de entrada de tipo `xs:date`.

6.6.16.17 month-from-dateTime

Devuelve un elemento `xs:integer` que representa la parte del mes del valor `xs:dateTime` dado como argumento.

Lenguajes

XQuery, XSLT 2.0.

Parámetros

Nombre	Tipo	Descripción
dateTime	`xs:dateTime`	El valor de entrada de tipo `xs:dateTime`.

6.6.16.18 months-from-duration

Devuelve un elemento `xs:integer` que representa el componente "meses" de la representación canónica del valor de duración dado como argumento.

Lenguajes

XQuery, XSLT 2.0.

Parámetros

Nombre	Tipo	Descripción
duration	`xs:duration`	El valor de entrada de tipo `xs:duration`.

6.6.16.19 seconds-from-dateTime

Devuelve un elemento `xs:integer` que representa la parte de los segundos del valor localizado de dateTime.

Lenguajes

XQuery, XSLT 2.0.

Parámetros

Nombre	Tipo	Descripción
dateTime	`xs:dateTime`	

6.6.16.20 seconds-from-duration

Devuelve un elemento `xs:integer` que representa el componente "segundos" de la la representación canónica del valor de duración dado como argumento.

Lenguajes

XQuery, XSLT 2.0.

Parámetros

Nombre	Tipo	Descripción
duration	`xs:duration`	El valor de entrada de tipo `xs:duration`.

6.6.16.21 seconds-from-time

Devuelve un elemento `xs:integer` que representa la parte del mes del valor `xs:time` dado como argumento.

Lenguajes

XQuery, XSLT 2.0.

Parámetros

Nombre	Tipo	Descripción
time	`xs:time`	El valor de entrada de tipo `xs:time`.

6.6.16.22 subtract-dateTimes

Devuelve el elemento `xs:dayTimeDuration` que corresponde a la diferencia entre el valor normalizado de **dateTime1** y el valor normalizado de **dateTime2**.

Lenguajes

XQuery, XSLT 2.0.

Parámetros

Nombre	Tipo	Descripción
dateTime1	`xs:dateTime`	El primer valor de entrada.

Nombre	Tipo	Descripción
dateTime2	`xs:dateTime`	El segundo valor de entrada.

6.6.16.23 subtract-dates

Devuelve el elemento `xs:dayTimeDuration` que corresponde a la diferencia entre el valor normalizado de **date1** y el valor normalizado de **date2**.

Lenguajes

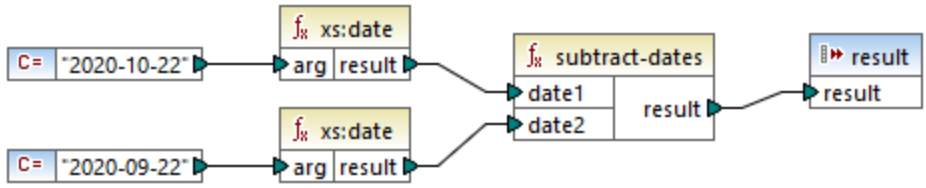
XQuery, XSLT 2.0.

Parámetros

Nombre	Tipo	Descripción
date1	`xs:date`	El primer valor de entrada.
date2	`xs:date`	El segundo valor de entrada.

Ejemplo

Esta asignación sustrae dos fechas (2020-10-22 menos 2020-09-22). El resultado es el valor **`P30D`** de tipo `xs:dayTimeDuration`., que representa una duración de 30 días.



6.6.16.24 subtract-times

Devuelve el elemento `xs:dayTimeDuration` que corresponde a la diferencia entre el valor normalizado de **time1** y el valor normalizado de **time2**.

Lenguajes

XQuery, XSLT 2.0.

Parámetros

Nombre	Tipo	Descripción
time1	`xs:time`	El primer valor de entrada.

Nombre	Tipo	Descripción
time2	`xs:time`	El segundo valor de entrada.

6.6.16.25 timezone-from-date

Devuelve el componente de huso horario de la fecha dada como argumento. El resultado es un componente `xs:dayTimeDuration` que indica una desviación de UTC; su valor puede ir de +14:00 horas a -14:00 horas, ambas incluidas.

Lenguajes

XQuery, XSLT 2.0.

Parámetros

Nombre	Tipo	Descripción
date	`xs:date`	El valor de entrada de tipo type `xs:date`.

6.6.16.26 timezone-from-dateTime

Devuelve el componente de huso horario del valor `xs:dateTime` dado como argumento. El resultado es un componente `xs:dayTimeDuration` que indica una desviación de UTC; su valor puede ir de +14:00 horas a -14:00 horas, ambas incluidas.

Lenguajes

XQuery, XSLT 2.0.

Parámetros

Nombre	Tipo	Descripción
dateTime	`xs:dateTime`	El valor de entrada de tipo `xs:dateTime`.

6.6.16.27 timezone-from-time

Devuelve el componente de huso horario del valor `xs:time` dado como argumento. El resultado es un componente `xs:dayTimeDuration` que indica una desviación de UTC; su valor puede ir de +14:00 horas a -14:00 horas, ambas incluidas.

Lenguajes

XQuery, XSLT 2.0.

Parámetros

Nombre	Tipo	Descripción
time	`xs:time`	El valor de entrada de tipo `xs:time`.

6.6.16.28 year-from-date

Devuelve un elemento `xs:integer` que representa la parte del año del valor `xs:date` dado como argumento.

Lenguajes

XQuery, XSLT 2.0.

Parámetros

Nombre	Tipo	Descripción
date	`xs:date`	El valor de entrada de tipo `xs:date`.

6.6.16.29 year-from-dateTime

Devuelve un elemento `xs:integer` que representa la parte del año del valor `xs:dateTime` dado como argumento.

Lenguajes

XQuery, XSLT 2.0.

Parámetros

Nombre	Tipo	Descripción
dateTime	`xs:dateTime`	El valor de entrada de tipo `xs:dateTime`.

6.6.16.30 years-from-duration

Devuelve un elemento `xs:integer` que representa el componente "años" de la representación léxica canónica del valor de duración dado como argumento.

Lenguajes

XQuery, XSLT 2.0.

Parámetros

Nombre	Tipo	Descripción
duration	`xs:duration`	El valor de entrada de tipo `xs:duration`.

6.6.17 xpath2 | node functions (nodo)

Las funciones de nodo de la biblioteca **xpath2** ofrecen información sobre los nodos (elementos) de un componente de asignación.

La función **lang** toma como argumento una cadena que identifica un código de idioma (como "en"). La función devuelve **true** o **false** dependiendo de si el nodo de contexto tiene un atributo `xml:lang` con el valor especificado por el argumento de la función.

Las funciones **local-name**, **name** y **namespace-uri** devuelven el nombre local, el nombre y el URI de espacio de nombres respectivamente del nodo de entrada. Por ejemplo, en el caso del nodo **altova:Products**, el nombre local es **Products**, el nombre es **altova:Products** y el URI de espacio de nombres es el URI del espacio de nombres al que está enlazado el prefijo altova: (consulte el ejemplo de la función **local-name** 346). Cada una de estas tres funciones tiene dos variantes:

- Sin argumento: la función se aplica al nodo de contexto (para ver un ejemplo del nodo de contexto consulte el ejemplo de la función **lang** 344).
- Con argumento (que debe ser un nodo): la función se aplica al nodo especificado.

La función **number** toma un nodo como entrada, atomiza el nodo (es decir, extrae su contenido), convierte el valor en un decimal y devuelve el valor convertido. Esta función tiene dos variantes:

- Sin argumento: la función se aplica al nodo de contexto (para ver un ejemplo del nodo de contexto consulte el ejemplo de la función **lang** 344).
- Con argumento (que debe ser un nodo): la función se aplica al nodo especificado.

6.6.17.1 lang

Devuelve **true** si el nodo de contexto tiene un atributo `xml:lang` con un valor que coincide exactamente con el argumento **testlang** o es un subconjunto del mismo. De lo contrario devuelve **false**.

Lenguajes

XQuery, XSLT 2.0.

Parámetros

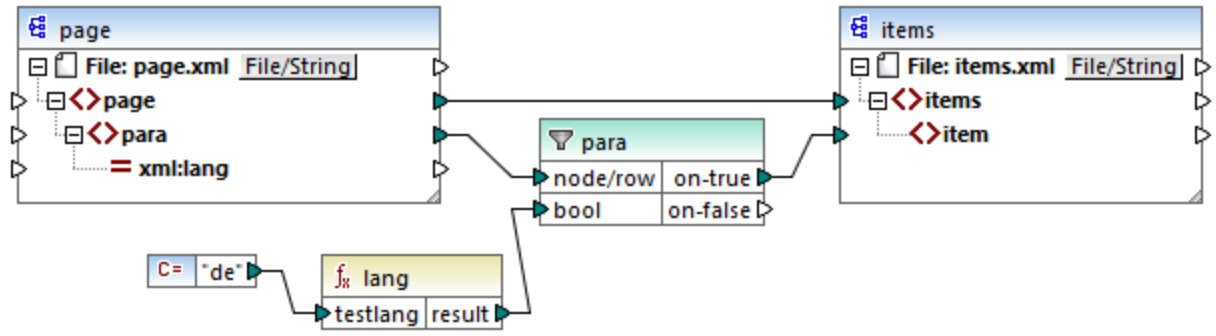
Nombre	Tipo	Descripción
testlang	`xs:string`	El código del idioma que se comprueba, por ejemplo "en".

Ejemplo

Este ejemplo XML contiene elementos **para** con distintos valores para el atributo `xml:lang`.

```
<page>
   <para xml:lang="en">Good day!</para>
   <para xml:lang="fr">Bonjour!</para>
   <para xml:lang="de-AT">Grüss Gott!</para>
   <para xml:lang="de-DE">Guten Tag!</para>
   <para xml:lang="de-CH">Grüezi!</para>
</page>
```

La asignación siguiente filtra solamente los párrafos en alemán, independientemente de la variante del país, gracias a la función **lang**.



Asignación XSLT 2.0

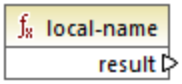
En la asignación anterior, por cada **para** de origen se crea un **item** en el componente de destino, de forma condicional. La condición la da un filtro que pasa al componente de destino sólo los nodos en los que la función **lang** devuelve **true**. Es decir, sólo los nodos cuyo atributo `xml:lang` es "de" (o un subconjunto de "de") cumplen la condición del filtro. En consecuencia, el resultado de la asignación es:

```
<items>
   <item>Grüss Gott!</item>
   <item>Guten Tag!</item>
   <item>Grüezi!</item>
</items>
```

Observe que la función **lang** opera en el contexto de cada **para** debido a la conexión primaria entre **para** and **item**, véase también Contexto de la asignación 409.

6.6.17.2 local-name

Devuelve la parte local del nombre del nodo de contexto como `xs:string`. Es una variante sin parámetros de la función **local-name** donde el nodo de contexto lo determinan las conexiones de la asignación. Para indicar un nodo de forma explícita use la función **local-name** 346 que toma un nodo de entrada como parámetro.

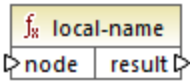


Lenguajes

XQuery, XSLT 2.0.

6.6.17.3 local-name

Devuelve la parte local del nombre del nodo como `xs:string`.



Lenguajes

XQuery, XSLT 2.0.

Parámetros

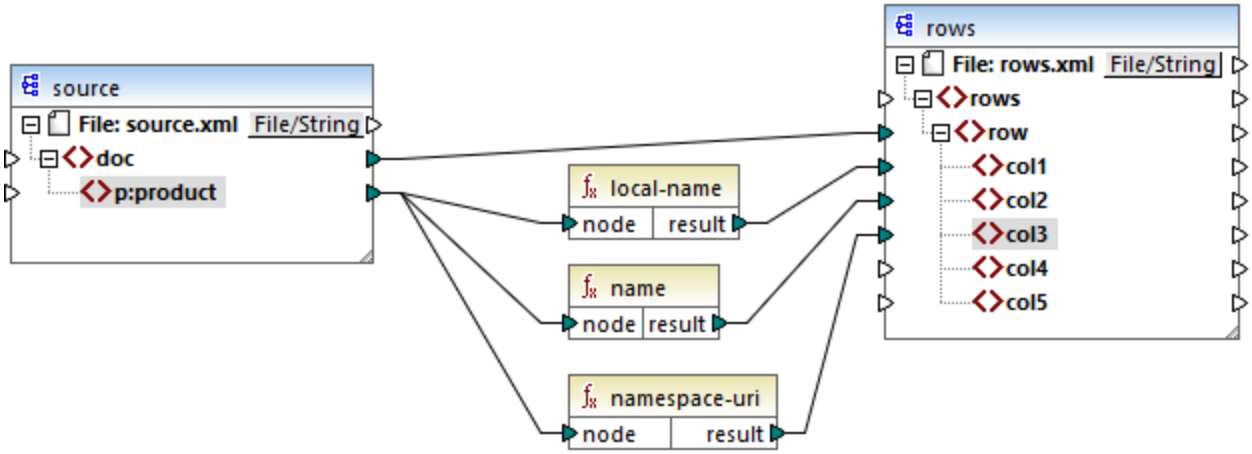
Nombre	Tipo	Descripción
node	**node()**	El nodo de entrada.

Ejemplo

En este archivo XML el nombre del elemento `p:product` es un nombre cualificado prefijado (QName). El prefijo "p" está asignado al espacio de nombres "http://mycompany.com".

```
<?xml version="1.0" encoding="UTF-8"?>
<doc xmlns:p="http://mycompany.com" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="source.xsd">
   <p:product/>
</doc>
```

Esta asignación extrae el nombre local, el nombre y el URI de espacio de nombres del nodo y escribe esos valores en un archivo de destino:



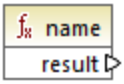
Asignación XSLT 2.0

A continuación puede ver el resultado de la asignación. Cada elemento **col** muestra el resultado de las funciones **local-name**, **name** y **namespace-uri**, respectivamente.

```
<rows>
   <row>
      <col1>product</col1>
      <col2>p:product</col2>
      <col3>http://mycompany.com</col3>
   </row>
</rows>
```

6.6.17.4 name

Devuelve el nombre del nodo de contexto. Es una variante sin parámetros de la función **name** donde el nodo de contexto lo determinan las conexiones de la asignación. Para indicar un nodo de forma explícita use la función **name** 348 que toma un nodo de entrada como parámetro.

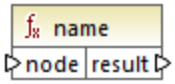


Lenguajes

XQuery, XSLT 2.0.

6.6.17.5 name

Devuelve el nombre de un nodo.



Lenguajes

XQuery, XSLT 2.0.

Parámetros

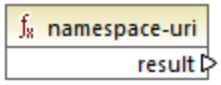
Nombre	Tipo	Descripción
node	`node()`	El nodo de entrada.

Ejemplo

Consulte el ejemplo de la función **`local-name`** 346.

6.6.17.6 namespace-uri

Devuelve el URI de espacio de nombres del QName del nodo de contexto, como `xs:string`. Es una variante sin parámetros de la función **`namespace-uri`** donde el nodo de contexto lo determinan las conexiones de la asignación. Para indicar un nodo de forma explícita use la función **`namespace-uri`** 349 que toma un nodo de entrada como parámetro.



Lenguajes

XQuery, XSLT 2.0.

6.6.17.7 namespace-uri

Devuelve el URI de espacio de nombres del QName del nodo como `xs:string`.

Lenguajes

XQuery, XSLT 2.0.

Parámetros

Nombre	Tipo	Descripción
node	**node()**	El nodo de entrada.

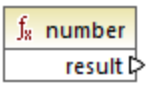
Ejemplo

Consulte el ejemplo de la función **local-name** 346.

6.6.17.8 number

Devuelve el valor del nodo de contexto convertido en `xs:double`. Es una variante sin parámetros de la función **number** donde el nodo de contexto lo determinan las conexiones de la asignación. Para indicar un nodo de forma explícita use la función **number** 349 que toma un nodo de entrada como parámetro.

Los únicos tipos que se pueden convertir en números son booleanos, cadenas numéricas y otros tipos numéricos. Los valores de entrada no numéricos (como cadenas no numéricas) resultan en NaN (Not a Number; *no es un número*).

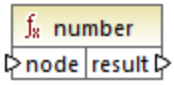


Lenguajes

XQuery, XSLT 2.0.

6.6.17.9 number

Devuelve el valor del nodo convertido en `xs:double`. Los únicos tipos que se pueden convertir en números son booleanos, cadenas numéricas y otros tipos numéricos. Los valores de entrada no numéricos (como cadenas no numéricas) resultan en NaN (Not a Number; *no es un número*).



Lenguajes

XQuery, XSLT 2.0.

Parámetros

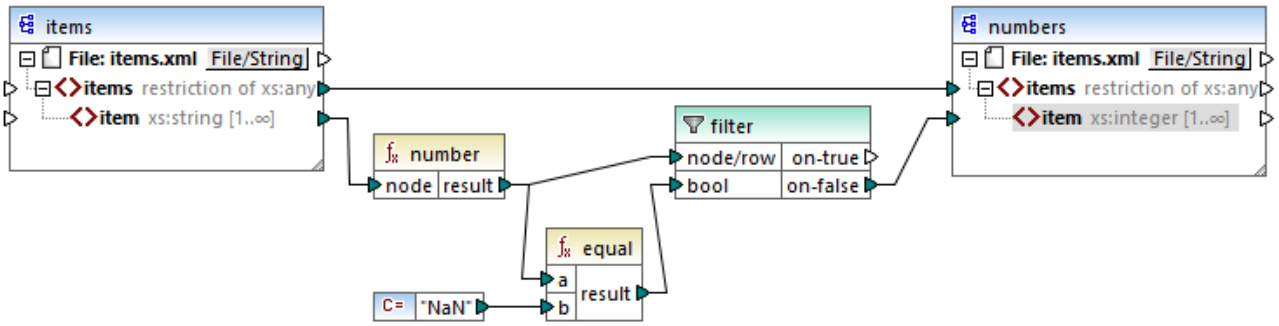
Nombre	Tipo	Descripción
node	`mf:atomic`	El nodo de entrada.

Ejemplo

Este XML contiene elementos de tipo `string`:

```
<items>
   <item>1</item>
   <item>2</item>
   <item>Jingle Bells</item>
</items>
```

La ilustración siguiente pretende convertir todas estas cadenas en valores numéricos y escribirlas en un archivo XML de destino. Observe que el tipo de datos del elemento **item** en el componente XML de destino es `xs:integer`, mientras que el elemento **item** de origen es de tipo `xs:string`. Si la conversión no se realiza correctamente, ese elemento se debe omitir y no se copiará en el archivo de destino.



Asignación XSLT 2.0

Para alcanzar el objetivo de la asignación se usó un filtro. La función **`equal`** comprueba si el resultado de la conversión es "NaN". Si no es así, indica que la conversión se realizó correctamente, por lo que el elemento sí se copia en el archivo de destino. El resultado de la asignación es el siguiente:

```
<items>
```

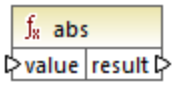
```
   <item>1</item>
   <item>2</item>
</items>
```

6.6.18 xpath2 | numeric functions (numéricas)

La biblioteca de funciones **xpath2 | numeric** ofrece las funciones numéricas `abs` y `round-half-to-even`.

6.6.18.1 abs

Devuelve el valor absoluto del argumento. Por ejemplo, si el argumento es **-2** o **2**, la función devuelve **2**.



Lenguajes

XQuery, XSLT 2.0.

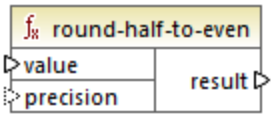
Parámetros

Nombre	Tipo	Descripción
value	`xs:decimal`	El valor de entrada.

6.6.18.2 round-half-to-even

La función `round-half-to-even` redondea el número indicado (primer argumento) con el grado de precisión (número de decimales) indicado en el segundo argumento. Por ejemplo, si el primer argumento es **2,141567** y el segundo es **3**, el número del primer argumento se redondea con tres decimales, por lo que el resultado es **2,141**. Si no se indica el segundo argumento el número se redondea sin decimales.

El elemento "even" de la función hace referencia al redondeo a un número par si el dígito indicado está entre dos valores. Por ejemplo, `round-half-to-even(3.475, 2)` devolvería **3,48**.



Lenguajes

XQuery, XSLT 2.0.

Parámetros

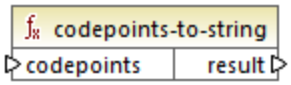
Nombre	Tipo	Descripción
value	`xs:decimal`	Obligatorio. Indica el valor de entrada que se redondea.
precision	`xs:integer`	Opcional. Indica el número de decimales del redondeo. El valor predeterminado es **0**.

6.6.19 xpath2 | string functions (cadena)

La biblioteca de funciones **xpath2 | string** permiten procesar cadenas (compararlas, convertirlas en mayúsculas o minúsculas, extraer subcadenas, etc.).

6.6.19.1 codepoints-to-string

Crea una cadena a partir de una secuencia de puntos de código Unicode. Esta función es opuesta a la función `string-to-codepoints` 360.



Lenguajes

XQuery, XSLT 2.0.

Parámetros

Nombre	Tipo	Descripción
codepoints	`ZeroOrMore xs:integer`	Esta entrada debe estar conectada a una secuencia de elementos de tipo integer, donde cada uno de ellos indica un punto de código Unicode.

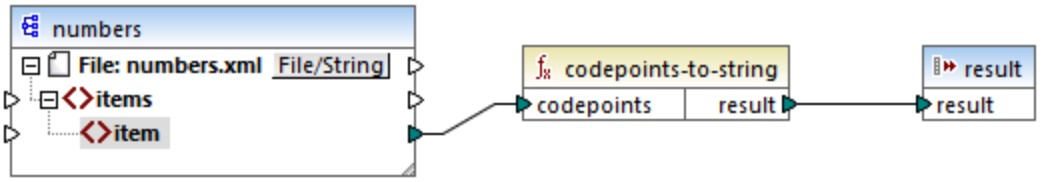
Ejemplo

Este XML contiene varios elementos **item** que almacenan valores de puntos de código Unicode.

```
<items>
   <item>77</item>
   <item>97</item>
```

```
   <item>112</item>
   <item>70</item>
   <item>111</item>
   <item>114</item>
   <item>99</item>
   <item>101</item>
</items>
```

Esta asignación suministra la secuencia de elementos como argumento a la función **codepoint-to-string**.



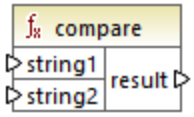
Asignación XSLT 2.0

El resultado de la función es **MapForce**.

6.6.19.2 compare

La función **compare** toma como argumentos dos cadenas y las compara. Si **string1** es alfabéticamente menor que **string2** (por ejemplo, las cadenas son "A" y "B") la función devuelve **-1**. Si son iguales ("A" y "A") la función devuelve **0**. Si **string1** es mayor que **string2** (por ejemplo "B" y "A"), entonces la función devuelve **1**.

Esta variante de las funciones usa el elemento collation predeterminado, que es Unicode. Existe otra variante 354 de esta función en la que puede usar el elemento collation como argumento.



Lenguajes

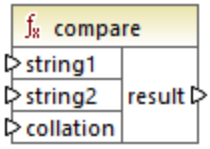
XQuery, XSLT 2.0.

Parámetros

Nombre	Tipo	Descripción
string1	xs:string	La primera cadena de entrada.
string2	xs:string	La segunda cadena de entrada.

6.6.19.3 compare

La función **compare** toma como argumentos dos cadenas y las compara usando el elemento collation dado como argumento. Si **string1** está más abajo en el alfabeto que **string2** (por ejemplo, las dos cadenas son "A" y "B"), entonces la función devuelve **-1**. Si las dos cadenas son iguales ("A" y "A") la función devuelve **0**. Si **string1** es mayor que **string2** (por ejemplo "B" y "A"), entonces la función devuelve **1**.



Lenguajes

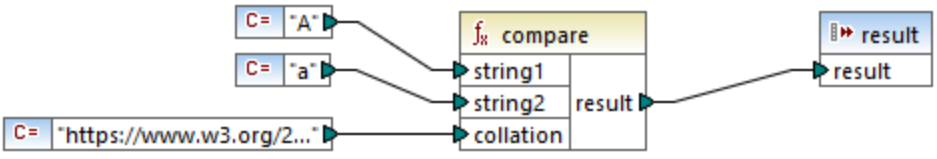
XQuery, XSLT 2.0.

Parámetros

Nombre	Tipo	Descripción
string1	`xs:string`	La primera cadena de entrada.
string2	`xs:string`	La primera cadena de entrada.
collation	`xs:string`	Indica qué cotejo usar para la comparación de cadenas. Puede provenir de la entrada de la función `default-collation` 326 o puede ser un cotejo como `http://www.w3.org/2005/xpath-functions/collation/html-ascii-case-insensitive`.

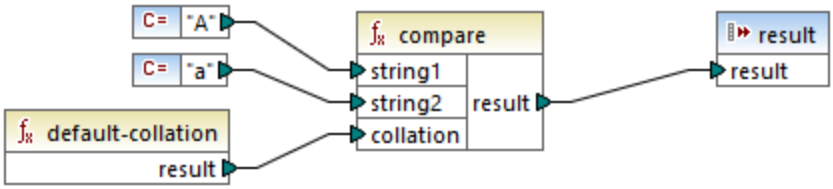
Ejemplo

Esta asignación compara las cadenas "A" y "a" usando el elemento collation que no distingue entre mayúsculas y minúsculas `http://www.w3.org/2005/xpath-functions/collation/html-ascii-case-insensitive`, que viene dado por una constante.



Asignación XSLT 2.0

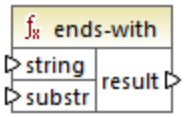
El resultado de la asignación anterior es **0** porque las dos se consideran iguales. Sin embargo, si reemplazamos el elemento collation con el que nos da la función **default-collation**, que usa los puntos de código Unicode predeterminados, el resultado se convierte en **-1** porque "A" es alfabéticamente menos que "a".



6.6.19.4 ends-with

Devuelve **true** si **string** termina en **substr**, de lo contrario devuelve **false**. El valor devuelto es de tipo `xs:boolean`.

Esta variante de la función usa el elemento collation predeterminado, que es Unicode. Existe otra variante 355 de esta función en la que puede usar el elemento collation como argumento.



Lenguajes

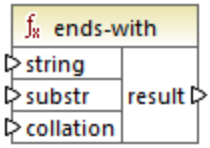
XQuery, XSLT 2.0.

Parámetros

Nombre	Tipo	Descripción
string	`xs:string`	La cadena de entrada (el "pajar").
substr	`xs:string`	La subcadena (la "aguja").

6.6.19.5 ends-with

Devuelve **true** si **string** termina en **substr**, de lo contrario devuelve **false**. El valor devuelto es de tipo `xs:boolean`.



Lenguajes

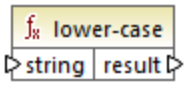
XQuery, XSLT 2.0.

Parámetros

Nombre	Tipo	Descripción
string	`xs:string`	La cadena de entrada (el "pajar").
substr	`xs:string`	La subcadena (la "aguja").
collation	`xs:string`	Indica qué cotejo usar para la comparación de cadenas. Puede provenir de la entrada de la función `default-collation`[326] o puede ser un cotejo como `http://www.w3.org/2005/xpath-functions/collation/html-ascii-case-insensitive`.

6.6.19.6 lower-case

Devuelve el valor de **string** después de pasar cada uno de sus caracteres a su versión en minúsculas.



Lenguajes

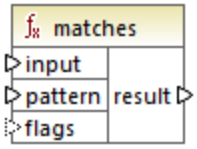
XQuery, XSLT 2.0.

Parámetros

Nombre	Tipo	Descripción
string	`xs:string`	El valor de entrada.

6.6.19.7 matches

La función **matches** comprueba si una cadena dada (el primer argumento) coincide con una expresión regular 8segundo argumento). La sintaxis de la expresión regular debe ser la definida por el aspecto `pattern` del esquema XML. La función devuelve **true** si la cadena coincide con la expresión regular; de lo contrario devuelve **false**.



Lenguajes

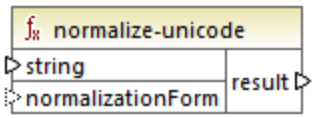
XQuery, XSLT 2.0.

Parámetros

Nombre	Tipo	Descripción
input	`xs:string`	La cadena de entrada.
pattern	`xs:string`	La expresión regular, véase Expresiones regulares 231.
flags	`xs:string`	Argumento opcional que influye en las coincidencias. Este argumento puede dar cualquier combinación de los elementos flag: `i`, `m`, `s`, `x`. Se pueden usar muchos flags, por ejemplo, `imx`. Si no se indica ninguno, se usan los valores predeterminados de los cuatro flags, que son: `i` Ignorar mayúsculas y minúsculas. El valor predeterminado es no ignorarlas. `m` Usar modo multilínea, en el que se considera que la cadena de entrada tiene varias líneas, cada una separada por un carácter de línea nueva (`x0a`). Los metacaracteres **^** u **$** indican el principio y final de cada línea. El modo predeterminado es de cadena, donde las cadenas empiezan y terminan con los caracteres **^** and **$**. `s` Usar el modo *dot-all*. El modo predeterminado es no usarlo, donde el metacarácter **.** encuentra todos los caracteres excepto el de línea nueva (`x0a`). En el modo *dot-all* el punto también encuentra la línea nueva. `x` Ignora los espacios en blanco. El modo predeterminado es no ignorarlos.

6.6.19.8 normalize-unicode

Devuelve el valor de **string** normalizado conforme a las reglas de normalización de la forma normal indicada (segundo argumento). Para más información sobre la normalización Unicode consulte §2.2 en https://www.w3.org/TR/charmod-norm/.



Lenguajes

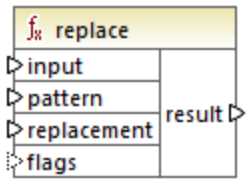
XQuery, XSLT 2.0.

Parámetros

Nombre	Tipo	Descripción
string	`xs:string`	El valor de cadena que se quiere normalizar.
normalizationForm	`xs:string`	El argumento opcional que da la forma normal. La predeterminada es la Unicode Normalization Form C (NFC). También son compatibles las formas normales NFC, NFD, NFKC y NFKD.

6.6.19.9 replace

Esta función toma como argumentos una cadena de entrada, una expresión regular y una cadena de reemplazo y reemplaza todas las coincidencias de la expresión regular en la cadena de entrada con la cadena de reemplazo. Si la expresión regular coincide con dos cadenas que se superponen en la cadena de entrada, sólo se reemplaza la primera.



Lenguajes

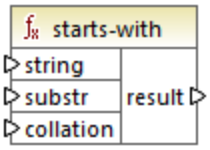
XQuery, XSLT 2.0.

Parámetros

Nombre	Tipo	Descripción
input	`xs:string`	La cadena de entrada.
pattern	`xs:string`	La expresión regular, véase Expresiones regulares 231.
replacement	`xs:string`	La cadena de reemplazo.
flags	`xs:string`	Argumento opcional que influye en las coincidencias. Se usa igual que el argumento **flags** de la función `matches` 357.

6.6.19.10 starts-with

Devuelve **true** si **string** empieza por **substr**; de lo contrario devuelve **false**. El valor devuelto es de tipo `xs:boolean`. La comparación de cadenas tiene lugar conforme al elemento collation que se indique.



Lenguajes

XQuery, XSLT 2.0.

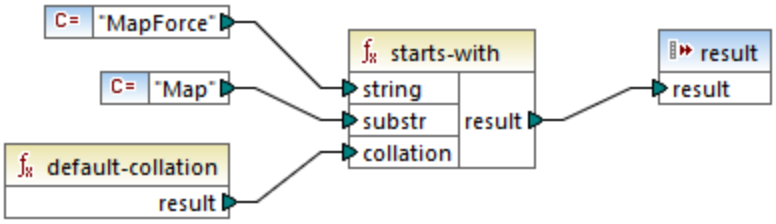
Parámetros

Nombre	Tipo	Descripción
string	`xs:string`	La cadena de entrada (el "pajar").
substr	`xs:string`	La subcadena (la "aguja").
collation	`xs:string`	Indica qué cotejo usar para la comparación de cadenas. Puede provenir de la entrada de la función `default-collation` 326 o puede ser un cotejo como `http://www.w3.org/2005/xpath`

Nombre	Tipo	Descripción
		`-functions/collation/html-ascii-case-insensitive`.

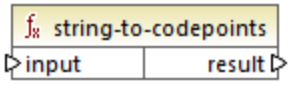
Ejemplo

The following mapping returns the value **`true`**, because the input string "MapForce" begins with the substring "Map", assuming that the default Unicode collation is used.



6.6.19.11 string-to-codepoints

Devuelve la secuencia de de puntos de código Unicode (valores enteros) que constituye la cadena dada como argumento. Esta función es opuesta a la función **`codepoints-to-string`**[352].



Lenguajes

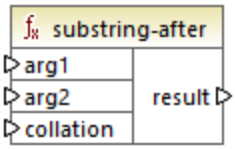
XQuery, XSLT 2.0.

Parámetros

Nombre	Tipo	Descripción
input	**`xs:string`**	La cadena de entrada.

6.6.19.12 substring-after

Devuelve la parte de **arg1** que aparece después de la cadena **arg2**.



Lenguajes

XQuery, XSLT 2.0.

Parámetros

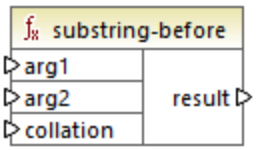
Nombre	Tipo	Descripción
arg1	`xs:string`	haystack
arg2	`xs:string`	haystack
collation	`xs:string`	Indica qué cotejo usar para la comparación de cadenas. Puede provenir de la entrada de la función `default-collation`[326] o puede ser un cotejo como `http://www.w3.org/2005/xpath-functions/collation/html-ascii-case-insensitive`.

Ejemplo

Si **arg1** es "MapForce", **arg2** es "Map" y **collation** es `default-collation`[326], la función devuelve "Force".

6.6.19.13 substring-before

Devuelve la parte de **arg1** que aparece antes de la cadena **arg2**.



Lenguajes

XQuery, XSLT 2.0.

Parámetros

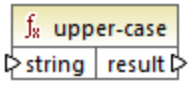
Nombre	Tipo	Descripción
arg1	`xs:string`	La cadena de entrada (el "pajar").
arg2	`xs:string`	La subcadena (la "aguja").
collation	`xs:string`	Indica qué cotejo usar para la comparación de cadenas. Puede provenir de la entrada de la función `default-collation`326 o puede ser un cotejo como `http://www.w3.org/2005/xpath-functions/collation/html-ascii-case-insensitive`.

Ejemplo

Si **arg1** es "MapForce", **arg2** es "Force" y **collation** es `default-collation`326, la función devuelve "Map".

6.6.19.14 upper-case

Devuelve el valor de **string** después de pasar cada uno de sus caracteres a su versión en mayúsculas.



Lenguajes

XQuery, XSLT 2.0.

Parámetros

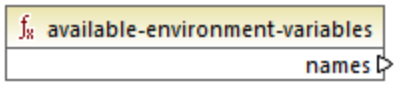
Nombre	Tipo	Descripción
string	`xs:string`	La cadena de entrada.

6.6.20 xpath3 | external information functions

Las funciones de información externas de la biblioteca **xpath3** permiten obtener información sobre el entorno de ejecución XSLT o recuperar datos de recursos externos.

6.6.20.1 available-environment-variables

Devuelve una lista de nombres de variables de entorno que se pueden pasar a la función **environment-variable** como una secuencia (puede que vacía) de cadenas.



Lenguajes

XSLT 3.0.

6.6.20.2 environment-variable

Devuelve el valor de una variable de entorno del sistema, si existe. El tipo de retorno es `xs: string`.

Lenguajes

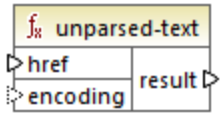
XSLT 3.0.

Parámetros

Nombre	Tipo	Descripción
name	**xs:string**	El nombre de la variable de entorno.

6.6.20.3 unparsed-text

Lee un recurso externo (por ejemplo un archivo) y devuelve una representación en cadena de ese recurso.



Lenguajes

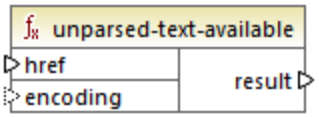
XSLT 3.0.

Parámetros

Nombre	Tipo	Descripción
href	`xs:string`	Una cadena en forma de referencia URI.
encoding	`xs:string`	Argumento opcional. Indica el nombre del tipo de cifrado, por ejemplo "UTF-8", "UTF-16". Si el cifrado no se puede determinar automáticamente se asume que es UTF-8.

6.6.20.4 unparsed-text-available

Determina si una llamada a **`unparsed-text`** con argumentos particulares se realiza correctamente. El tipo de retorno es `xs:boolean`.



Lenguajes

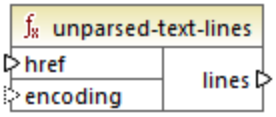
XSLT 3.0.

Parámetros

Nombre	Tipo	Descripción
href	`xs:string`	Una cadena en forma de referencia URI.
encoding	`xs:string`	Argumento opcional. Indica el nombre del tipo de cifrado, por ejemplo "UTF-8", "UTF-16". Si el cifrado no se puede determinar automáticamente se asume que es UTF-8.

6.6.20.5 unparsed-text-lines

Lee un recurso externo (por ejemplo un archivo) y devuelve su contenido como una secuencia de cadenas, una por cada línea de texto en la representación de las cadenas del recurso.



Lenguajes

XSLT 3.0.

Parámetros

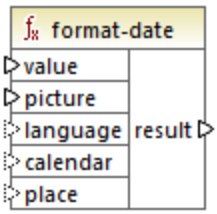
Nombre	Tipo	Descripción
href	`xs:string`	Una cadena en forma de referencia URI.
encoding	`xs:string`	Argumento opcional. Indica el nombre del tipo de cifrado, por ejemplo "UTF-8", "UTF-16". Si el cifrado no se puede determinar automáticamente se asume que es UTF-8.

6.6.21 xpath3 | formatting functions

Las funciones de formato de la biblioteca **xpath3** se usan para dar formato a los valores date, time e integer.

6.6.21.1 format-date

Devuelve una cadena que contiene un valor `xs:date` con formato para su visualización.



Lenguajes

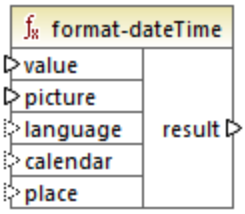
XSLT 3.0.

Parámetros

Nombre	Tipo	Descripción
value	`xs:date`	El valor de entrada `xs:date` al que se debe aplicar formato. Parámetro obligatorio.
picture	`xs:string`	Parámetro obligatorio. Consulte el apartado 9.8.4.1 de la recomendación "XPath and XQuery Functions and Operators 3.1" W3C Recommendation (https://www.w3.org/TR/xpath-functions-31) para obtener más información.
language	`xs:string`	Parámetro opcional. Consulte el apartado 9.8.4.8 de la recomendación "XPath and XQuery Functions and Operators 3.1" W3C Recommendation (https://www.w3.org/TR/xpath-functions-31) para obtener más información.
calendar	`xs:string`	Véase el punto anterior.
place	`xs:string`	Véase el punto anterior.

6.6.21.2 format-dateTime

Devuelve una cadena que contiene un valor `xs: dateTime` con formato para su visualización.



Lenguajes

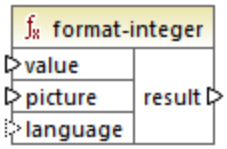
XSLT 3.0.

Parámetros

Nombre	Tipo	Descripción
value	`xs:dateTime`	El valor de entrada `xs:dateTime` al que se debe aplicar formato.
picture	`xs:string`	Parámetro obligatorio. Consulte el apartado 9.8.4.1 de la recomendación "XPath and XQuery Functions and Operators 3.1" W3C Recommendation (https://www.w3.org/TR/xpath-functions-31) para obtener más información.
language	`xs:string`	Parámetro opcional. Consulte el apartado 9.8.4.8 de la recomendación "XPath and XQuery Functions and Operators 3.1" W3C Recommendation (https://www.w3.org/TR/xpath-functions-31) para obtener más información.
calendar	`xs:string`	Véase el punto anterior.
place	`xs:string`	Véase el punto anterior.

6.6.21.3 format-integer

Da formato a un número entero conforme a una cadena de imagen dada y usando las convenciones del lenguaje natural indicado.



Lenguajes

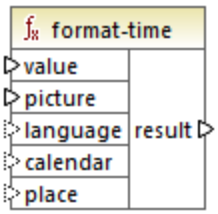
XSLT 3.0.

Parámetros

Nombre	Tipo	Descripción
value	`xs:integer`	El valor del número entero de entrada al que se debe aplicar formato.
picture	`xs:string`	Parámetro obligatorio. Consulte el apartado 4.6.1 de la recomendación "XPath and XQuery Functions and Operators 3.1" W3C Recommendation (https://www.w3.org/TR/xpath-functions-31) para obtener más información.
language	`xs:string`	Parámetro opcional. Indica el lenguaje natural según el cual se aplica formato al valor. Si se indica, este valor debe ser una cadena vacía o cualquier valor permitido para el atributo `xml:lang` de acuerdo con la recomendación "Extensible Markup Language (XML) 1.0 W3C Recommendation (https://www.w3.org/TR/xml) para obtener más información.

6.6.21.4 format-time

Devuelve una cadena que contiene un valor `xs: time` con formato para su visualización.



Lenguajes

XSLT 3.0.

Parámetros

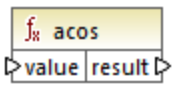
Nombre	Tipo	Descripción
value	`xs:time`	El valor de entrada `xs: time` al que se debe aplicar formato.
picture	`xs:string`	Parámetro obligatorio. Consulte el apartado 9.8.4.1 de la recomendación "XPath and XQuery Functions and Operators 3.1" W3C Recommendation (https://www.w3.org/TR/xpath-functions-31) para obtener más información.
language	`xs:string`	Parámetro opcional. Consulte el apartado 9.8.4.8 de la recomendación "XPath and XQuery Functions and Operators 3.1" W3C Recommendation (https://www.w3.org/TR/xpath-functions-31) para obtener más información.
calendar	`xs:string`	Véase el punto anterior.
place	`xs:string`	Véase el punto anterior.

6.6.22 xpath3 | math functions

Las funciones matemáticas de la biblioteca **xpath3** se usan para realizar cálculos trigonométricos, entre otros.

6.6.22.1 acos

El resultado es el arco coseno de un ángulo en el rango de **0** a **pi**.



Lenguajes

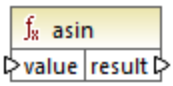
XSLT 3.0.

Parámetros

Nombre	Tipo	Descripción
value	`xs:double`	El valor de entrada.

6.6.22.2 asin

El resultado es el arcoseno de un ángulo en el rango de **-pi/2** a **pi/2**.



Lenguajes

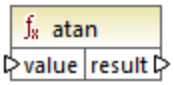
XSLT 3.0.

Parámetros

Nombre	Tipo	Descripción
value	`xs:double`	El valor de entrada.

6.6.22.3 atan

El resultado es el arco tangente de un ángulo en el rango de **-pi/2** a **pi/2**.



Lenguajes

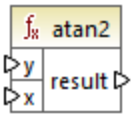
XSLT 3.0.

Parámetros

Nombre	Tipo	Descripción
value	`xs:double`	El valor de entrada.

6.6.22.4 atan2

Devuelve el ángulo en radianes subtendido en el origen por el punto de un plano con coordenadas (x, y) y el eje x positivo.



Lenguajes

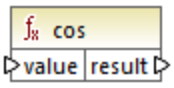
XSLT 3.0.

Parámetros

Nombre	Tipo	Descripción
y	`xs:double`	La coordenada x.
x	`xs:double`	La coordenada y.

6.6.22.5 cos

El resultado es el coseno del ángulo del parámetro de entrada. La unidad del valor es el radián.



Lenguajes

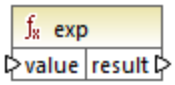
XSLT 3.0.

Parámetros

Nombre	Tipo	Descripción
value	`xs:double`	El valor de entrada.

6.6.22.6 exp

El resultado es *e* (el logaritmo natural base) elevado a la potencia de la entrada value.



Lenguajes

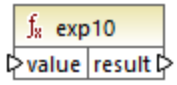
XSLT 3.0.

Parámetros

Nombre	Tipo	Descripción
value	`xs:double`	El valor de entrada.

6.6.22.7 exp10

El resultado es 10 elevado a la potencia de la entrada value.



Lenguajes

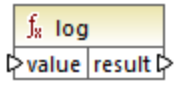
XSLT 3.0.

Parámetros

Nombre	Tipo	Descripción
value	`xs:double`	El valor de entrada.

6.6.22.8 log

El resultado es el logaritmo natural (en base e) de un valor.



Lenguajes

XSLT 3.0.

Parámetros

Nombre	Tipo	Descripción
value	`xs:double`	El valor de entrada.

6.6.22.9 log10

El resultado es el logaritmo decimal (en base 10) de un valor.

Lenguajes

XSLT 3.0.

Parámetros

Nombre	Tipo	Descripción
value	`xs:double`	El valor de entrada.

6.6.22.10 pi

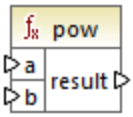
Devuelve una aproximación a la constante matemática **pi**.

Lenguajes

XSLT 3.0.

6.6.22.11 pow

El resultado es el valor de **a** elevado a la potencia **b**.



Lenguajes

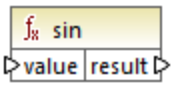
XSLT 3.0.

Parámetros

Nombre	Tipo	Descripción
a	`xs:double`	El valor de entrada **a**.
b	`xs:double`	El valor de entrada **b**.

6.6.22.12 sin

El resultado es el seno del ángulo del parámetro de entrada. La unidad del valor es el radián.



Lenguajes

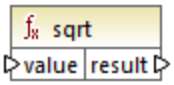
XSLT 3.0.

Parámetros

Nombre	Tipo	Descripción
value	`xs:double`	El valor de entrada.

6.6.22.13 sqrt

Devuelve la raíz cuadrada no negativa del argumento.



Lenguajes

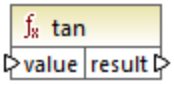
XSLT 3.0.

Parámetros

Nombre	Tipo	Descripción
value	`xs:double`	El valor de entrada.

6.6.22.14 tan

El resultado es la tangente del ángulo del parámetro de entrada. La unidad del valor es el radián.



Lenguajes

XSLT 3.0.

Parámetros

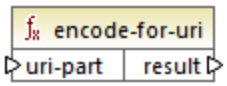
Nombre	Tipo	Descripción
value	`xs:double`	El valor de entrada.

6.6.23 xpath3 | URI functions

Las funciones de URI de la biblioteca **xpath3** cifran, convierten y aplican secuencias de escape a los valores que se quieren usar en URIs.

6.6.23.1 encode-for-uri

Cifra caracteres reservados en una cadena que se pretende usar en el segmento de la ruta de un URI. Para más información sobre esta función consulte el apartado 6.2 de la recomendación "XPath and XQuery Functions and Operators 3.1" W3C Recommendation (https://www.w3.org/TR/xpath-functions-31) para obtener más información.



Lenguajes

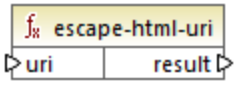
XSLT 3.0.

Parámetros

Nombre	Tipo	Descripción
uri-part	`xs:string`	El valor del URI de entrada al que se debe aplicar el cifrado.

6.6.23.2 escape-html-uri

Aplica secuencias de escape a un URI de igual forma que los agentes de usuario HTML administran los valores de atributo que se espera que contengan los URIs. Para más información sobre esta función consulte el apartado 6.4 de la recomendación "XPath and XQuery Functions and Operators 3.1" W3C Recommendation (https://www.w3.org/TR/xpath-functions-31) para obtener más información.



Lenguajes

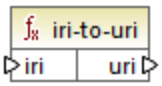
XSLT 3.0.

Parámetros

Nombre	Tipo	Descripción
uri	`xs:string`	El valor del URI de entrada al que se deben aplicar las secuencias de escape.

6.6.23.3 iri-to-uri

Convierte una cadena que contiene un IRI (Identificador de recursos internacionalizado) en un URI (Identificador de recursos uniforme). Para más información sobre esta función consulte el apartado 6.3 de la recomendación "XPath and XQuery Functions and Operators 3.1" W3C Recommendation (https://www.w3.org/TR/xpath-functions-31) para obtener más información.



Lenguajes

XSLT 3.0.

Parámetros

Nombre	Tipo	Descripción
iri	`xs:string`	El valor de IRI de entrada.

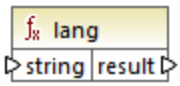
6.6.24 xslt | xpath functions

Las funciones de la biblioteca xslt | xpath son funciones de conjunto de nodos XPath 1.0. Estas funciones toman un nodo o conjunto de nodos como contexto y devuelve información sobre ellos. Las funciones `last` y `position` operan en el contexto de asignación actual 409, que determinan las conexiones de la asignación.

Nota: en la biblioteca de funciones **core** puede encontrar más funciones XPath 1.0.

6.6.24.1 lang

Devuelve **true** si el nodo de contexto tiene un atributo `xml:lang` con un valor que o coincide totalmente con el argumento **string** o es un subconjunto de esta. De lo contrario, la función devuelve **false**.



Lenguajes

XSLT 1.0.

Parámetros

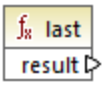
Nombre	Tipo	Descripción
string	`xs:string`	El código del idioma que se comprueba, por ejemplo "en".

Ejemplo

Consulte el ejemplo de la función `lang` 344 de la biblioteca **xpath2**.

6.6.24.2 last

Devuelve el número de la posición del último nodo en la lista de nodos que se procesa.



Lenguajes

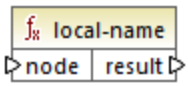
XSLT 1.0.

Ejemplo

Consulte el ejemplo de la función `last` 327 de la biblioteca **xpath2**.

6.6.24.3 local-name

Devuelve la parte local del nombre del nodo indicado como argumento.



Lenguajes

XSLT 1.0, XSLT 2.0.

Parámetros

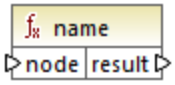
Nombre	Tipo	Descripción
node	`node()`	El nodo de entrada.

Ejemplo

Consulte el ejemplo de la función `local-name` 346 de la biblioteca **xpath2**.

6.6.24.4 name

Devuelve el nombre del nodo dado como argumento.



Lenguajes

XSLT 1.0, XSLT 2.0.

Parámetros

Nombre	Tipo	Descripción
node	`node()`	El nodo de entrada.

Ejemplo

Consulte el ejemplo de la función `local-name` 346 de la biblioteca **xpath2**

6.6.24.5 namespace-uri

Devuelve el URI de espacio de nombres del nodo dado como argumento.

Lenguajes

XSLT 1.0, XSLT 2.0.

Parámetros

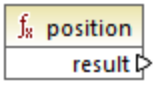
Nombre	Tipo	Descripción
node	`node()`	El nodo de entrada.

Ejemplo

Consulte el ejemplo de la función **local-name** 346 de la biblioteca **xpath2**

6.6.24.6 position

Devuelve la posición del nodo actual en el conjunto de nodos que se está procesando.



Lenguajes

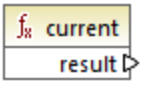
XSLT 1.0.

6.6.25 xslt | xslt functions

Las funciones de esta biblioteca con funciones XSLT 1.0.

6.6.25.1 current

Esta función no toma argumentos y devuelve el nodo actual.

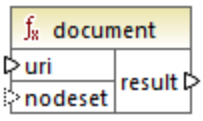


Lenguajes

XSLT 1.0.

6.6.25.2 document

Accede a los nodos desde un documento externo. El resultado de la función se envía a un nodo del documento de destino.



Lenguajes

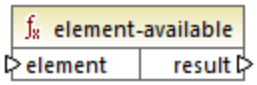
XSLT 1.0.

Parámetros

Nombre	Tipo	Descripción
uri	`xs:string`	Obligatorio. Indica la ruta al documento de destino. El documento XML debe ser válido y poder analizarse.
nodeset	`nodo()`	Opcional. Indica un nodo, el URI de base que se usa para resolver el URI indicado como primer argumento si es relativo.

6.6.25.3 element-available

Esta función comprueba si un elemento es compatible con el procesador XSLT (el elemento se da como único argumento de cadena de la función). La cadena del argumento se evalúa como QName Por tanto, los elementos XSLT deben tener un prefijo `xsl:` y los elementos XML Schema deben tener un prefijo `xs:` porque estos los prefijos declarados para estos espacios de nombres en el XSLT que se genera para la asignación. La función devuelve un valor booleano.



Lenguajes

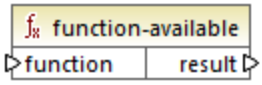
XSLT 1.0.

Parámetros

Nombre	Tipo	Descripción
element	`xs:string`	El nombre del elemento.

6.6.25.4 function-available

Esta función es similar a la función **`element-available`** y comprueba si el nombre de la función dado como argumento es compatible con el procesador XSLT. La cadena de entrada se evalúa como QName. La función devuelve un valor booleano.



Lenguajes

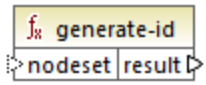
XSLT 1.0.

Parámetros

Nombre	Tipo	Descripción
Función	`xs:string`	El nombre de la función.

6.6.25.5 generate-id

Esta función genera una cadena única que identifica el primer nodo del conjunto de nodos identificado por el argumento de entrada opcional. Si no tiene ningún argumento, la función genera el identificador en el nodo de contexto. El resultado se puede dirigir después a cualquier nodo del documento de salida.



Lenguajes

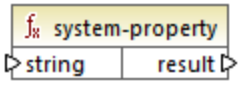
XSLT 1.0, XSLT 2.0, XSLT 3.0.

Parámetros

Nombre	Tipo	Descripción
nodeset	`nodo()`	Argumento opcional que indica el nodo de entrada.

6.6.25.6 system-property

Esta función devuelve las propiedades del procesador XSLT (el sistema). Para los procesadores XSLT hay tres propiedades de sistema obligatorias, todas situadas en el espacio de nombres XSLT. Se trata de `xsl:version`, `xsl:vendor` y `xsl:vendor-url`. La cadena de entrada se evalúa como QName y debe tener el prefijo `xsl:` porque este es el prefijo asociado con el espacio de nombres XSLT de la hoja de estilos XSLT subyacente.



Lenguajes

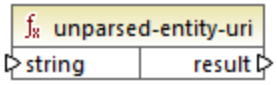
XSLT 1.0, XSLT 2.0, XSLT 3.0.

Parámetros

Nombre	Tipo	Descripción
string	`xs:string`	Indica el nombre de la propiedad, que puede ser cualquiera de estos: `xsl:version`, `xsl:vendor`, `xsl:vendor-url`.

6.6.25.7 unparsed-entity-uri

Si usa una DTD puede declarar en ella una entidad sin analizar. Esta entidad sin analizar (por ejemplo, una imagen) tendrá un URI que ubica la entidad sin analizar. La cadena de entrada de la función debe coincidir con el nombre de la entidad sin analizar que se declaró en la DTD. La función devuelve el URI de la entidad sin analizar, que puede dirigirse a cualquier nodo del documento de destino (al nodo **href** por ejemplo).



Lenguajes

XSLT 1.0.

Parámetros

Nombre	Tipo	Descripción
string	`xs:string`	El nombre de la entidad sin analizar cuyo URI se quiere obtener.

7 Asignaciones avanzadas

Sitio web de Altova: Herramienta de integración de datos

En esta sección explicamos los distintos tipos de asignaciones avanzadas y se incluyen los apartados siguientes:

7.1 Asignar nombres de nodos

En la mayoría de las asignaciones de datos de MapForce el objetivo es leer *valores* de un componente de origen y escribir *valores* en un componente de destino. Sin embargo, en algunos casos el objetivo no es sólo acceder a los valores de los nodos de origen, sino al nombre de los nodos. Por ejemplo, puede tratarse de una asignación que lea el nombre de los elementos o atributos (y no sus valores) de un archivo XML de origen y convertir esos nombres en valores de elemento o atributo en el XML de destino.

Imaginemos que tenemos un archivo XML que contiene una lista de productos. Cada producto tiene este formato:

```
<product>
   <id>1</id>
   <color>red</color>
   <size>10</size>
</product>
```

Nuestro objetivo es convertir información sobre cada producto en pares nombre/valor. Por ejemplo:

```
<product>
   <attribute name="id" value="1" />
   <attribute name="color" value="red" />
   <attribute name="size" value="10" />
</product>
```

En este caso, necesitaremos acceder al nombre de los nodos de forma *dinámica* para realizar conversiones de datos.

Nota: La transformación del ejemplo también puede conseguirse con las funciones **node-name** 275 y **static-node-name** 276 de la biblioteca **core**. No obstante, en nuestro ejemplo necesitamos conocer exactamente los nombres de elemento del componente de origen y necesitamos conectar cada uno de los elementos manualmente con el componente de destino. Además, puede que las funciones mencionadas no sean suficiente para filtrar o agrupar nodos por nombre o para manipular el tipo de datos de los nodos.

No sólo puede acceder al nombre de los nodos de forma dinámica cuando necesite leer nombres de nodo, sino también cuando necesite escribirlos. En una asignación normal y corriente el nombre de los atributos o elementos del componente de destino siempre se conoce antes de que se ejecute la asignación. Estos nombres de atributo o elemento vienen del esquema subyacente del componente. Sin embargo, con los nombres de nodo dinámicos podrá crear atributos o elementos nuevos cuyo nombre no se conoce antes de que se ejecute la asignación. Concretamente, el nombre del atributo o elemento viene dado por la asignación propiamente dicha (por el componente de origen).

Para poder acceder a elementos o atributos secundarios de forma dinámica el nodo debe tener elementos o atributos secundarios y el nodo principal no puede ser el nodo XML raíz.

Puede trabajar con nombres de nodo dinámicos si en la asignación de datos participan componentes de este tipo:

- XML
- CSV/FLF*

* Sólo compatible con las ediciones Professional o Enterprise de MapForce.

Puede trabajar con nombres de nodo dinámicos si selecciona estos lenguajes de transformación: motor integrado BUILT-IN*, XSLT2, XQuery*, C#*, C++*, Java*.

* Estos lenguajes sólo son compatibles con MapForce Professional o Enterprise Edition.

Para más información consulte el apartado Obtener acceso al nombre de los nodos 386 o el ejemplo del apartado Ejemplo: asignar nombres de elemento a valores de atributo 398.

7.1.1 Obtener acceso al nombre de los nodos

Cuando un nodo de un componente XML tiene nodos secundarios, podemos obtener tanto el nombre como el valor de cada nodo secundario en la asignación directamente. Esta técnica se denomina *nombres de nodo dinámicos.* El adjetivo *dinámicos* hace referencia al hecho de que el procesamiento se realiza *sobre la marcha,* durante la ejecución de la asignación, y no se basa en la información estática del esquema que se conoce antes de que se ejecute la asignación. En este apartado explicamos cómo habilitar el acceso dinámico a nombres de nodo y cómo utilizarlo.

Cuando se leen datos de un componente de origen, los *nombres de nodo dinámicos* permiten:

- Obtener una lista de nodos o atributos secundarios de un nodo en forma de secuencia. En MapForce una *secuencia* es una lista de cero o más elementos que se pueden conectar con un componente de destino y que crea tantos elementos en el componente de destino como elementos existen en el componente de origen. Por ejemplo, si un nodo tiene cinco atributos en el componente de origen, puede crear cinco elementos nuevos en el componente de destino (cada uno de ellos corresponderá a un atributo).
- Leer no sólo los valores de los nodos secundarios (como hacen las asignaciones normales y corrientes) sino también sus nombres.

Cuando se escriben datos en un componente de destino, los *nombres de nodo dinámicos* permiten:

- Crear nodos nuevos usando nombres que vienen dados por la asignación (nombres *dinámicos*), en lugar de los nombres que vienen dados por la configuración del componente (nombres *estáticos*).

A continuación veremos un ejemplo basándonos en el esquema XML **<Documentos>\Altova\MapForce2024\MapForceExamples\Tutorial\Products.xsd**. Este esquema viene acompañado de un documento de instancia llamado **Products.xml**. Para agregar tanto el esquema como el archivo de instancia al área de asignación seleccione **Insertar | Archivo o esquema XML** y navegue hasta el archivo **<Documentos>\Altova\MapForce2024\MapForceExamples\Tutorial\Products.xml**. Cuando la aplicación solicite un elemento raíz, seleccione `products`.

Para habilitar los nombres de nodo dinámicos para el nodo `product` haga clic con el botón derecho en ese nodo y seleccione uno de estos comandos en el menú contextual:

- **Mostrar atributos con nombre dinámico** si desea tener acceso a los atributos del nodo o

- **Mostrar elementos secundarios con nombre dinámico** si desea tener acceso a los elementos secundarios del nodo.

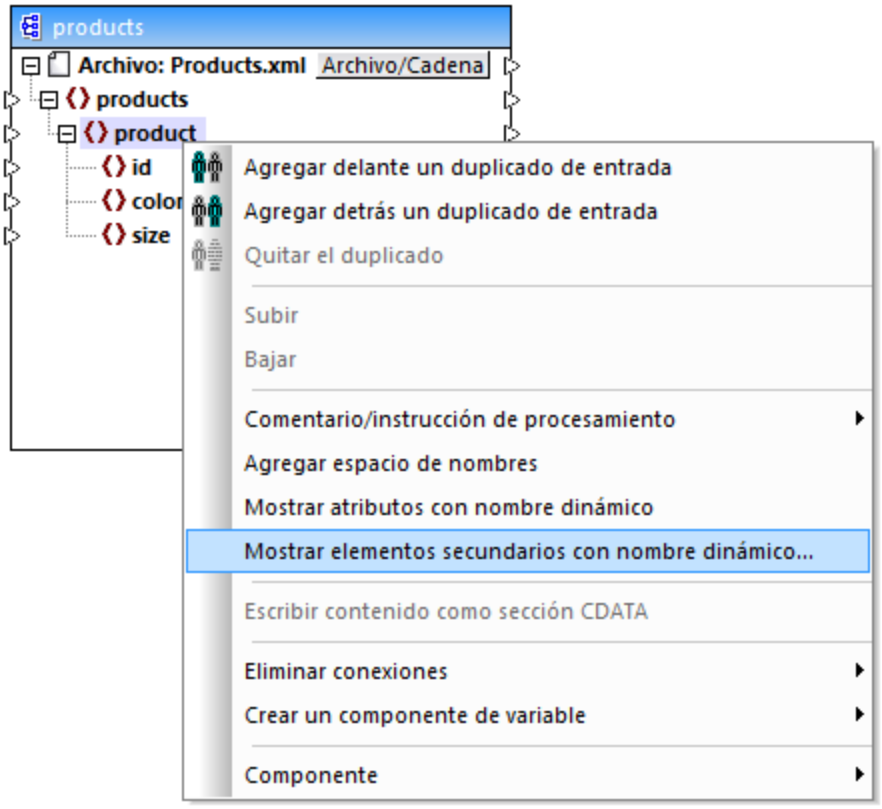


Fig. 1: Habilitar nombres de nodo dinámicos (para elementos secundarios)

Nota: Los comandos que aparecen en la imagen anterior solamente están disponibles cuando se trata de nodos con nodos secundarios. Además, estos comandos no están disponibles cuando se trata del nodo raíz.

Cuando se cambia un nodo al modo dinámico, aparece un cuadro de diálogo como el de la imagen siguiente. Siguiendo con nuestro ejemplo, ahora configuramos las opciones que puede ver en la imagen (estas opciones se definen en detalle en el apartado Obtener acceso a determinado tipo de nodos 393).

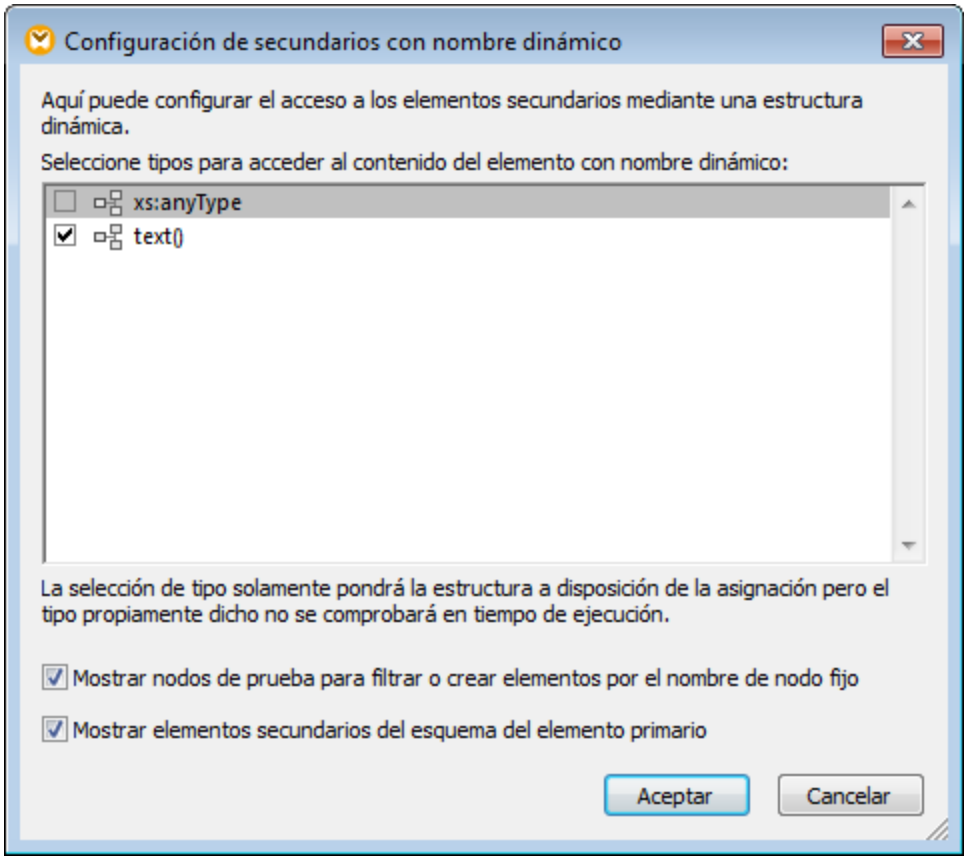


Fig. 2: Cuadro de diálogo "Configuración de secundarios con nombre dinámico"

A continuación explicamos lo que hace el componente cuando se habilitan los nombres de nodo dinámicos para el nodo `product`. Observe que el aspecto del componente cambia significativamente.

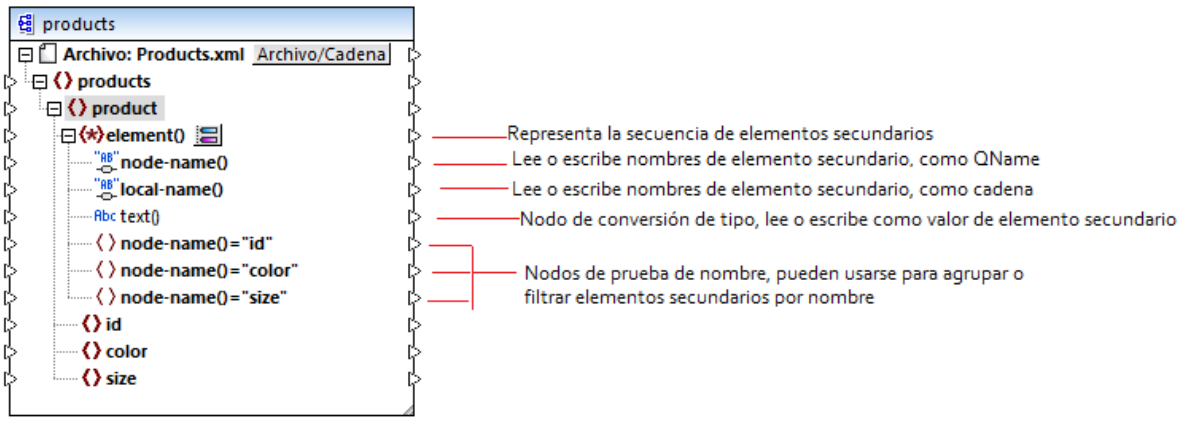


Fig. 3: Nombres de nodo dinámicos habilitados (para elementos secundarios)

Para restaurar el modo estándar del componente haga clic con el botón derecho en el nodo `product` y desactive el comando **Mostrar elementos secundarios con nombre dinámico** en el menú contextual.

En la imagen siguiente podemos ver el aspecto que tiene el componente cuando se habilita el acceso dinámico a los atributos de un nodo (haciendo clic con el botón secundario en el elemento `product` y seleccionando el comando **Mostrar atributos con nombre dinámico** en el menú contextual).

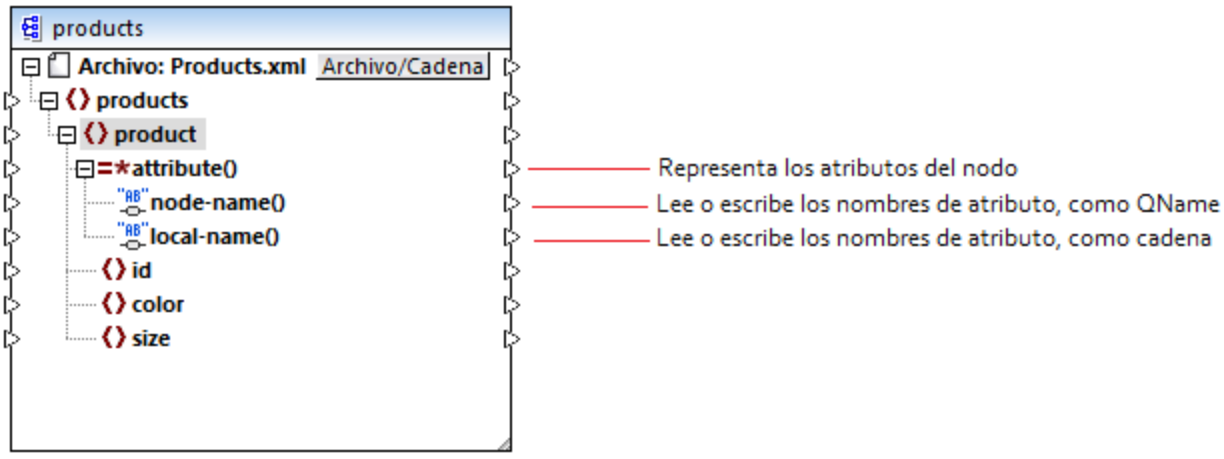


Fig. 4: Nombres de nodo dinámicos habilitados (para atributos)

Para restaurar el modo estándar del componente haga clic con el botón derecho en el nodo `product` y desactive el comando **Mostrar atributos con nombre dinámico** en el menú contextual.

Como puede ver en las dos imágenes anteriores, el aspecto del componente cambia cuando se habilita el modo *nombres de nodo dinámicos* en uno de los nodos (en nuestro ejemplo se trata del nodo `product`). El nuevo aspecto del componente ofrece varias opciones nuevas porque permite:

- Leer o escribir una lista con todos los elementos secundarios o atributos de un nodo (que vienen dados por el elemento `element()` o `attribute()` respectivamente).
- Leer o escribir el nombre de cada elemento secundario o atributo (que viene dado por el elemento `node-name()` o `local-name()` respectivamente).
- Si se trata de elementos, leer o escribir el valor de cada elemento secundario, en el tipo de datos que corresponda. Este valor viene dado por el nodo de conversión de tipo (el elemento `text()`). Recuerde que solamente los elementos pueden tener nodos de conversión de tipo. Los atributos siempre tienen el tipo "string".
- Agrupar o filtrar elementos secundarios por nombre.

A continuación explicamos con qué tipos de nodos puede trabajar cuando use el modo *nombres de nodo dinámicos.*

element()

Este nodo diferentes comportamientos, dependiendo de si está en el componente de origen o en el componente de destino. Si está en el componente de origen, aporta los elementos secundarios del nodo en forma de secuencia. En la Fig. 3 (ver más arriba) el elemento `element()` ofrece una lista (secuencia) con todos los elementos secundarios de `product`. Por ejemplo, la secuencia que se crea a partir de este XML contendría tres elementos (porque `product` tiene tres elementos secundarios):

```
<product>
```

```
    <id>1</id>
    <color>red</color>
    <size>10</size>
  </product>
```

Observe que el nombre y el tipo de cada elemento de la secuencia es el que nos da el nodo `node-name()` y el nodo de conversión de tipo respectivamente. Imagine que necesita pasar datos de un archivo XML a otro XML de destino de la siguiente manera:

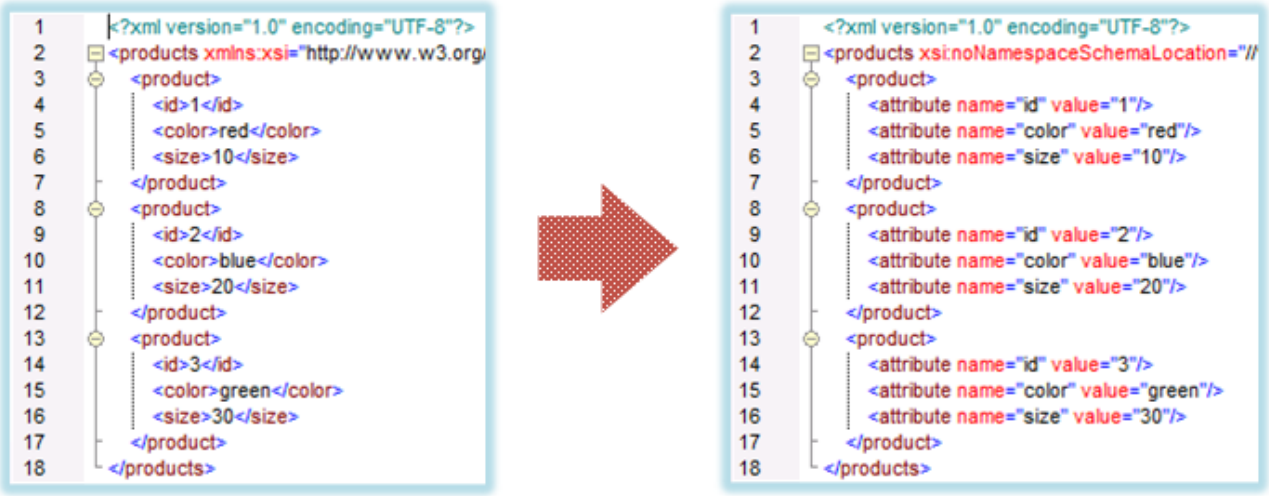


Fig. 6: Asignación entre nombres de elemento XML y valores de atributo

Para conseguir este objetivo debemos diseñar esta asignación de datos:

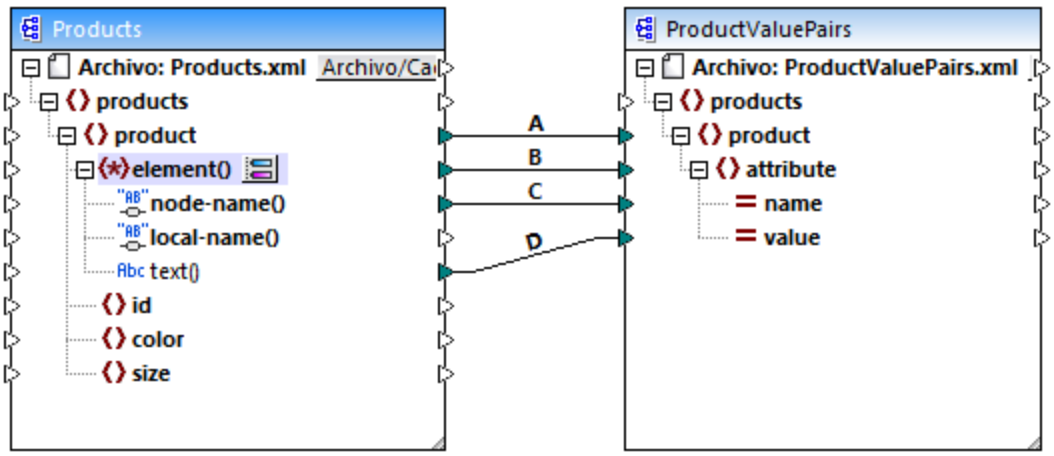


Fig. 7: Asignar nombres de elemento XML a valores de atributo

El papel que `element()` desempeña aquí es aportar la secuencia de elementos secundarios de `product`, mientras que `node-name()` y `text()` aportan el nombre y el valor real de cada elemento de la secuencia. Esta

asignación se describe más detalladamente en el apartado Ejemplo: asignar nombres de elemento a valores de atributo 398.

En el componente de destino `element()` no crea nada por sí mismo, lo cual constituye una excepción de la regla básica de asignación (por cada elemento del origen, crear un elemento de destino). Los elementos propiamente dichos los crean los nodos de conversión de tipo (usando el valor de `node-name()`) y por los nodos de prueba de nombre (usando su propio nombre).

attribute()

Como puede verse en la figura nº4, este elemento permite acceder a todos los atributos del nodo en tiempo de ejecución de la asignación. En un componente de origen aporta, en forma de secuencia, los atributos del nodo de origen que está conectado. Por ejemplo, en este archivo XML, la secuencia incluiría dos elementos (porque `product` tiene dos atributos):

```
<product id="1" color="red" />
```

Observe que el nodo `attribute()` sólo aporta el valor de cada atributo en la secuencia, siempre como tipo string. El nombre de cada atributo viene dado por el nodo `node-name()`.

En un componente de destino este nodo procesa una secuencia conectada y crea un valor de atributo por cada elemento de la secuencia. El nombre de atributo viene dado por `node-name()`. Por ejemplo, imagine que necesita pasar datos de un archivo XML a otro archivo XML de la siguiente manera:

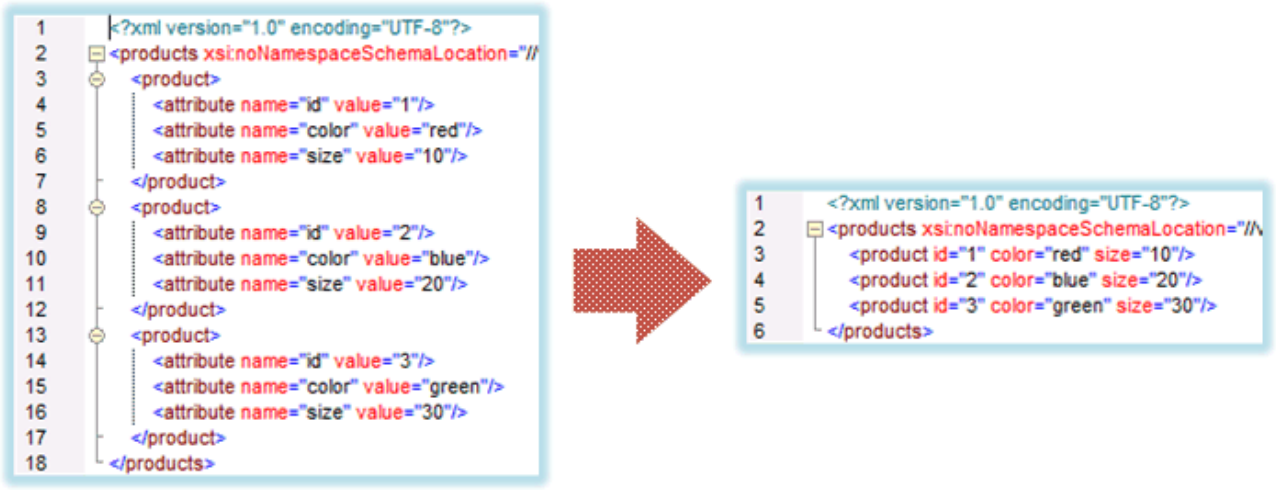


Fig. 8: Asignar valores a nombres de atributo

Para conseguir este objetivo debemos diseñar esta asignación de datos:

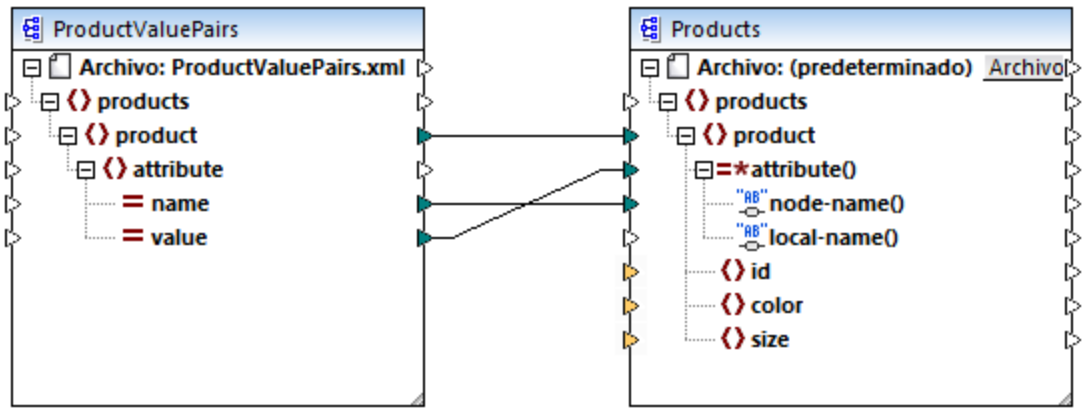


Fig. 9: Asignar valores de atributo a nombres de atributo

Nota: Esta transformación puede conseguirse sin habilitar el acceso dinámico a los atributos del nodo, pero aquí usamos el acceso dinámico para explicar cómo funciona `attribute()` en un componente de destino.

Si quiere reconstruir esta asignación, tenga en cuenta que usa los mismos componentes XML que la asignación **ConvertProducts.mfd** de la carpeta **<Documentos>\Altova\MapForce2024\MapForceExamples\Tutorial**. La única diferencia es que el destino ahora es el origen y el origen ahora es el destino. Como datos de entrada del componente de origen deberá utilizar una instancia XML que contenga valores de atributo, por ejemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<products>
   <product>
      <attribute name="id" value="1"/>
      <attribute name="color" value="red"/>
      <attribute name="size" value="big"/>
   </product>
</products>
```

Nota: Para simplificar el ejemplo se omitió la declaración de espacio de nombres y de esquema en el fragmento de código anterior.

node-name()

En un componente de origen `node-name()` aporta el nombre de cada elemento secundario de `element()` o el nombre de cada atributo de `attribute()` respectivamente. Por defecto el nombre que aporta `node-name()` es de tipo `xs:QName`. Para obtener el nombre como tipo string debe utilizarse el nodo `local-name()` (*véase la figura nº3*).

En un componente de destino `node-name()` escribe el nombre de cada elemento o atributo que exista en `element()` o `attribute()`.

local-name()

Este nodo funciona igual que el nodo `node-name()`, pero su tipo es `xs:string` en lugar de `xs:QName`.

Nodo de conversión de tipo

En un componente de origen el nodo de conversión de tipo aporta el valor de cada elemento secundario que incluye `element()`. El nombre y la estructura de este nodo dependerá del tipo qeu esté seleccionado en el cuadro de diálogo "Configuración de secundarios con nombre dinámico" (figura nº2).

Para cambiar el tipo del nodo haga clic en el botón **Cambiar selección** y seleccione un tipo de la lista de tipos disponibles, incluido el comodín de esquema (`xs:any`). Consulte Obtener acceso a determinado tipo de nodos 393 para obtener más información.

En un componente de destino el nodo de conversión de tipo escribe el valor de cada elemento secundario que incluye `element()`, en el tipo de datos correspondiente. El tipo de datos deseado se selecciona haciendo clic en el botón **Cambiar selección**.

Nodos de prueba de nombre

En un componente de origen los nodos de prueba de nombre permiten agrupar o filtrar elementos secundarios de una instancia de origen por nombre. Por ejemplo, si necesita filtrar elementos secundarios por nombre para que la asignación acceda a los datos de instancia usando el tipo correcto (*véase* *Obtener acceso a determinado tipo de nodos 393*).

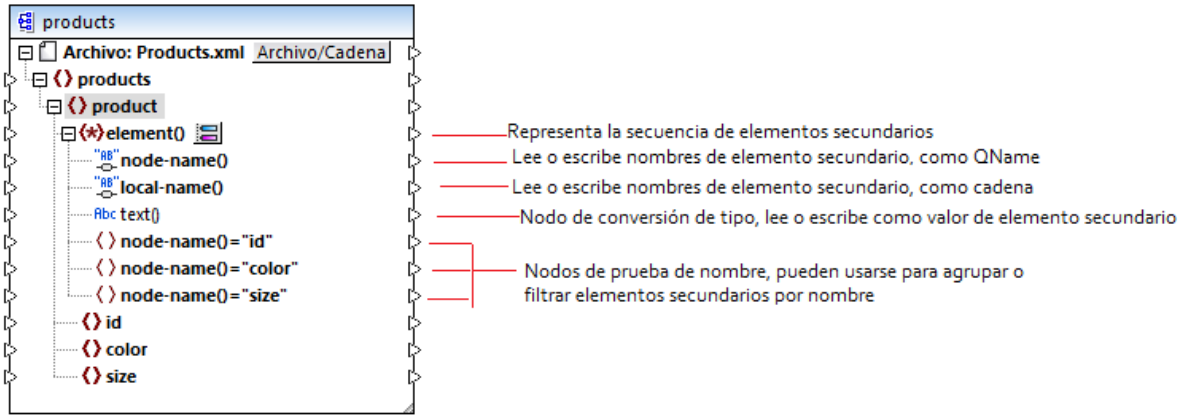
En general los nodos de prueba de nombre funcionan casi igual que los nodos de elemento normales para leer y escribir valores y subestructuras. Sin embargo, como la semántica de la asignación es distinta cuando está habilitado el acceso dinámico, existen algunas restricciones. Por ejemplo, no se puede concatenar el valor de dos nodos de prueba de nombre.

En un componente de destino los nodos de prueba de nombre crean tantos elementos en el resultado como elementos existen en la secuencia de origen conectada. Su nombre reemplaza el valor que esté asignado a `node-name()`.

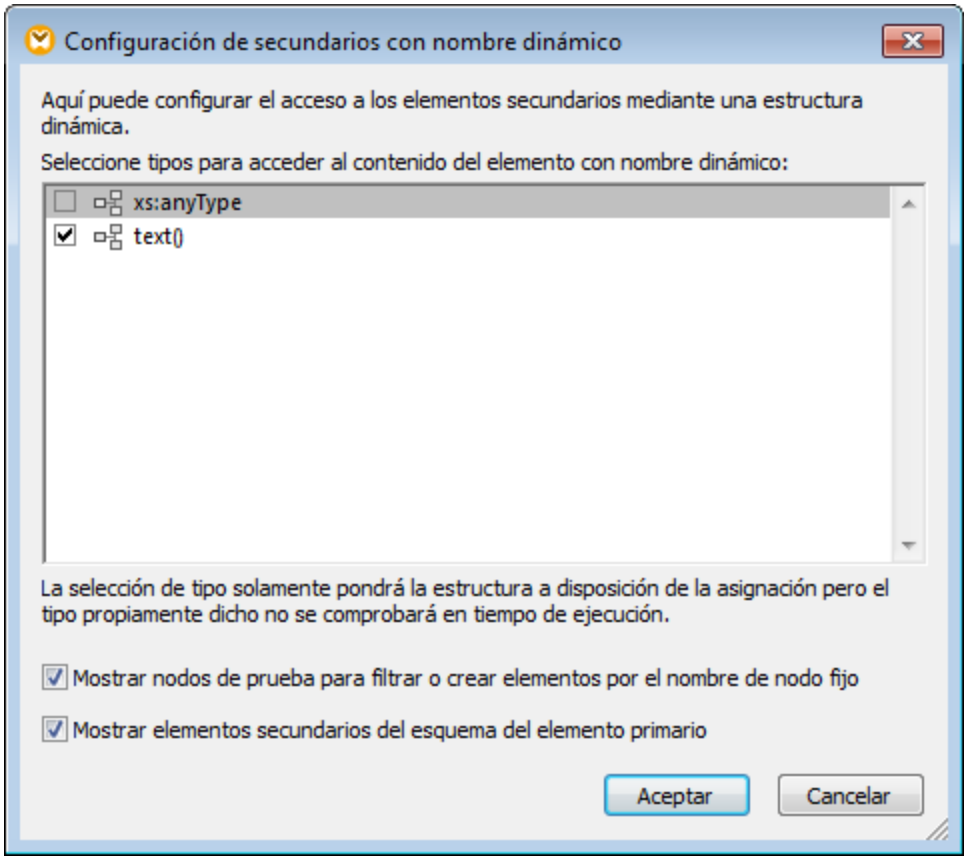
Si es necesario, puede ocultar los nodos de prueba de nombre en el componente. Esto se hace haciendo clic con el botón **Cambiar selección** del nodo `element()` y desactivando la casilla *Mostrar nodos de prueba...* en el cuadro de diálogo "Configuración de secundarios con nombre dinámico" (figura nº2).

7.1.2 Obtener acceso a determinado tipo de nodos

Como decíamos en el apartado anterior, podemos acceder a todos los elementos secundarios de un nodo haciendo clic con el botón derecho en el nodo y seleccionando el comando **Mostrar elementos secundarios con nombre dinámico** en el menú contextual. En tiempo de ejecución esto permitirá acceder al nombre de cada elemento secundario a través del nodo `node-name()`, mientras que el valor estará disponible a través de un nodo de conversión de tipo especial. En la imagen siguiente podemos ver que el nodo de conversión de tipo es el nodo `text()`.



Es importante señalar que el tipo de datos de cada elemento secundario no se conoce antes de que se ejecute la asignación. Además, el tipo de datos de cada elemento secundario puede ser distinto. Por ejemplo, un nodo `product` del archivo de instancia XML puede tener un elemento secundario `id` de tipo `xs:integer` y un elemento secundario `size` de tipo `xs:string`. Para poder acceder al contenido del nodo de un tipo específico el cuadro de diálogo (*imagen siguiente*) se abre cada vez que se habilita el acceso dinámico a los elementos secundarios de un nodo. Este cuadro de diálogo se puede abrir en cualquier momento con el botón **Cambiar selección** situado junto al nodo `element()`.



Cuadro de diálogo "Configuración de secundarios con nombre dinámico"

Para acceder al contenido de cada elemento en tiempo de ejecución tenemos varias opciones:

1. Acceder al contenido como cadena: marque la casilla **text()** del cuadro de diálogo. En este caso, se crea un nodo `text()` en el componente cuando se cierra el cuadro de diálogo. Esta opción es la más adecuada si el contenido es de tipo simple (`xs:int`, `xs:string`, etc.) y se describe en el apartado Ejemplo: asignar nombres de elemento a valores de atributo 398. Recuerde que el nodo **text()** solamente aparece si un nodo secundario del nodo actual puede contener texto.
2. Acceder al contenido como cierto tipo complejo permitido por el esquema. Cuando el esquema permite para el nodo seleccionado tipos complejos personalizados definidos globalmente, éstos también estarán disponibles en el cuadro de diálogo y podremos marcar sus casillas. En la imagen anterior, por ejemplo, no hay tipos complejos definidos globalmente por el esquema así que no hay ninguno en el cuadro de diálogo.
3. Acceder al contenido como cualquier tipo. Esto puede ser muy práctico en asignaciones de datos complejas y se consigue marcando la casilla de **xs:anyType**.

Debe tener en cuenta que en tiempo de ejecución MapForce no dispone de información (a través del nodo de conversión de tipo) sobre el tipo real del nodo de instancia. Por tanto, su asignación debe acceder al contenido del nodo usando el tipo correcto. Por ejemplo, si se espera que el nodo de una instancia XML de origen tenga nodos secundarios de varios tipos complejos, deberá hacer lo siguiente:
a) Asignar al nodo de conversión de tipo el tipo complejo que debe corresponderle.

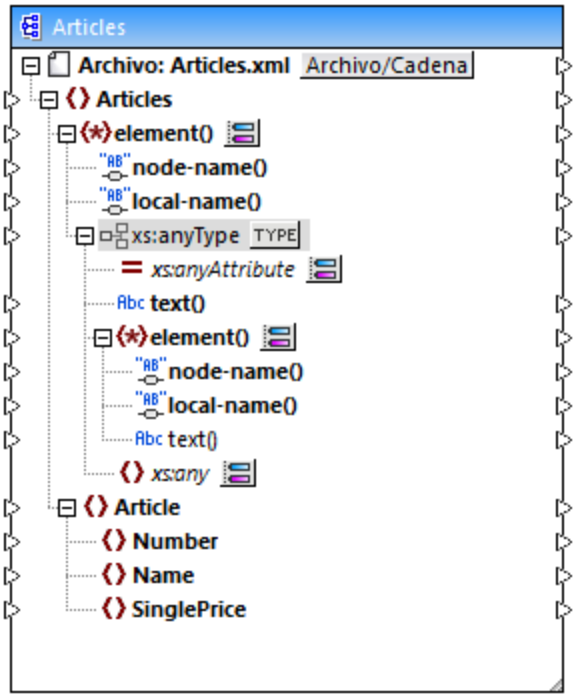
b) Agregar un filtro para leer de la instancia solamente el tipo complejo que debe corresponder.

Para más información sobre los filtros consulte Filtros y condiciones 178.

Acceder a estructuras más profundas

En asignaciones más complejas es posible acceder a nodos de niveles más profundos del esquema que el nivel de secundarios inmediatos de un nodo. En las asignaciones más sencillas como las del apartado Ejemplo: asignar nombres de elemento a valores de atributo 398 esta técnica no es necesaria porque la asignación sólo accede a los secundarios inmediatos de un nodo XML. Sin embargo, cuando lo necesite también podrá acceder de forma dinámica a estructuras más profundas (p.ej. a *nietos*, *bisnietos*, etc.) si sigue estos pasos:

1. Cree una asignación nueva.
2. Seleccione el comando de menú **Insertar | Archivo o esquema XML** y navegue hasta el archivo XML de instancia (p.ej. **Articles.xml** de la carpeta **<Documentos>\Altova\MapForce2024\MapForceExamples\Tutorial**).
3. Haga clic con el botón derecho en el nodo `Articles` y seleccione **Mostrar elementos secundarios con nombre dinámico** en el menú contextual.
4. Marque la casilla **xs:anyType** en el cuadro de diálogo "Configuración de secundarios con nombre dinámico".
5. Haga clic con el botón derecho en el nodo `xs:anyType` y seleccione otra vez **Mostrar elementos secundarios con nombre dinámico** en el menú contextual.
6. Marque la casilla **text()** en el cuadro de diálogo "Configuración de secundarios con nombre dinámico".



En la imagen puede ver que el componente tiene dos nodos `element()`. El segundo de ellos ofrece acceso dinámico a los nietos del nodo `<Articles>` de la instancia **Articles.xml**.

```
<?xml version="1.0" encoding="UTF-8"?>
<Articles xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="Articles.xsd">
   <Article>
      <Number>1</Number>
      <Name>T-Shirt</Name>
      <SinglePrice>25</SinglePrice>
   </Article>
   <Article>
      <Number>2</Number>
      <Name>Socks</Name>
      <SinglePrice>2.30</SinglePrice>
   </Article>
   <Article>
      <Number>3</Number>
      <Name>Pants</Name>
      <SinglePrice>34</SinglePrice>
   </Article>
   <Article>
      <Number>4</Number>
      <Name>Jacket</Name>
      <SinglePrice>57.50</SinglePrice>
   </Article>
</Articles>
```

Articles.xml

Por ejemplo, para obtener los nombres de elementos *nietos* (`Number`, `Name`, `SinglePrice`) debemos dibujar una conexión entre el nodo `local-name()` del segundo nodo `element()` y un elemento de destino. Igualmente para obtener los valores de elementos *nietos* (`1`, `T-Shirt`, `25`), debemos dibujar una conexión desde el nodo `text()`.

Aunque en este ejemplo no es relevante, MapForce ofrece la posibilidad de seguir habilitando nombres de nodo dinámico para los siguientes nodos `xs:anyType` y así poder alcanzar niveles aún más profundos de la estructura.

Debe tener en cuenta que:

- El botón TYPE sirve para seleccionar cualquier tipo derivado del esquema actual y mostrarlo en un nodo independiente. Esto puede ser práctico a la hora de crear asignaciones entre tipos de esquema derivados (*véase* *Tipos XML Schema derivados* 119).
- El botón **Cambiar selección** situado junto a un nodo `element()` abre el cuadro de diálogo "Configuración de secundarios con nombre dinámico".
- El botón **Cambiar selección** situado junto a `xs:anyAttribute` permite seleccionar cualquier atributo definido globalmente en el esquema. Igualmente el botón **Cambiar selección** situado junto al elemento `xs:any` permite seleccionar cualquier elemento definido globalmente en el esquema. Se trata del mismo funcionamiento de las asignaciones de comodines de esquema (*véase* *Comodines: xs:any / xs:anyAttribute* 125). Si usa esta opción, asegúrese de que el esquema permite que el atributo o elemento exista de verdad.

7.1.3 Ejemplo: asignar nombres de elemento a valores de atributo

Este ejemplo demuestra cómo asignar nombres de elemento de un documento XML a valores de atributo de un documento XML de destino. El ejemplo viene acompañado del diseño de asignación **<Documentos>\Altova\MapForce2024\MapForceExamples\Tutorial\ConvertProducts.mfd**.

Para comprender cómo funciona el ejemplo primero debemos imaginar que tenemos un archivo XML que contiene una lista de productos. Cada producto tiene este formato:

```
<product>
   <id>1</id>
   <color>red</color>
   <size>10</size>
</product>
```

Nuestro objetivo es convertir la información disponible sobre cada producto en pares de nombre/valor. Por ejemplo:

```
<product>
   <attribute name="id" value="1" />
   <attribute name="color" value="red" />
   <attribute name="size" value="10" />
</product>
```

Para conseguirlo el diseño de asignación del ejemplo utiliza la característica de MapForce conocida como "acceso dinámico a nombres de nodo". El término *dinámico* hace referencia al hecho de que, cuando se ejecute la asignación, se podrán leer los nombres de nodo (no sólo los valores) y estos nombres se podrán usar como valores. A continuación explicamos cómo diseñar esta asignación.

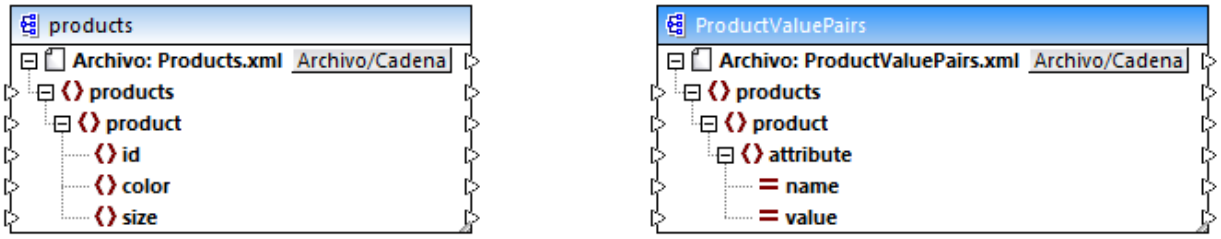
Paso nº1: agregar el componente XML de origen a la asignación

- En el menú **Insertar** haga clic en el comando **Archivo o esquema XML** y navegue hasta el archivo **<Documentos>\Altova\MapForce2024\MapForceExamples\Tutorial\Products.xml**. Este archivo XML apunta al esquema **Products.xsd** que está situado en la misma carpeta.

Paso nº2: agregar el componente XML de destino a la asignación

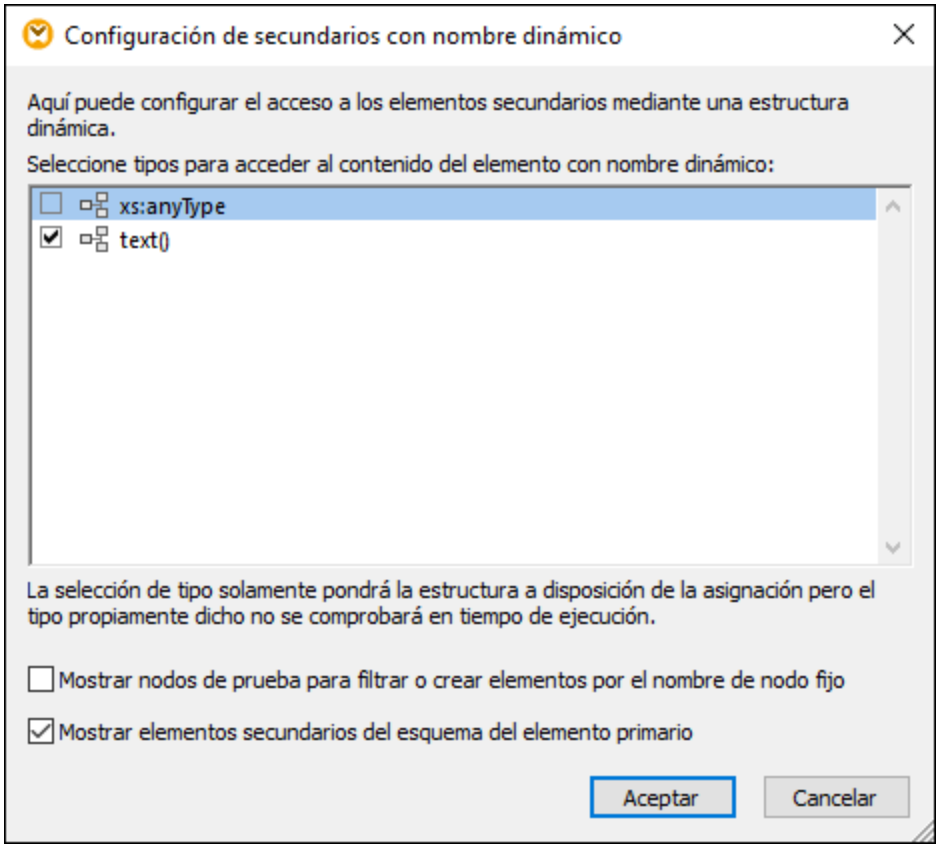
- En el menú **Insertar** haga clic en el comando **Archivo o esquema XML** y navegue hasta el archivo de esquema **<Documentos>\Altova\MapForce2024\MapForceExamples\Tutorial\ProductValuePairs.xsd**. Cuando la aplicación solicite un archivo de instancia, haga clic en **Omitir**. Cuando solicite un elemento raíz, seleccione `products`.

Llegados a este punto el diseño tendrá este aspecto:

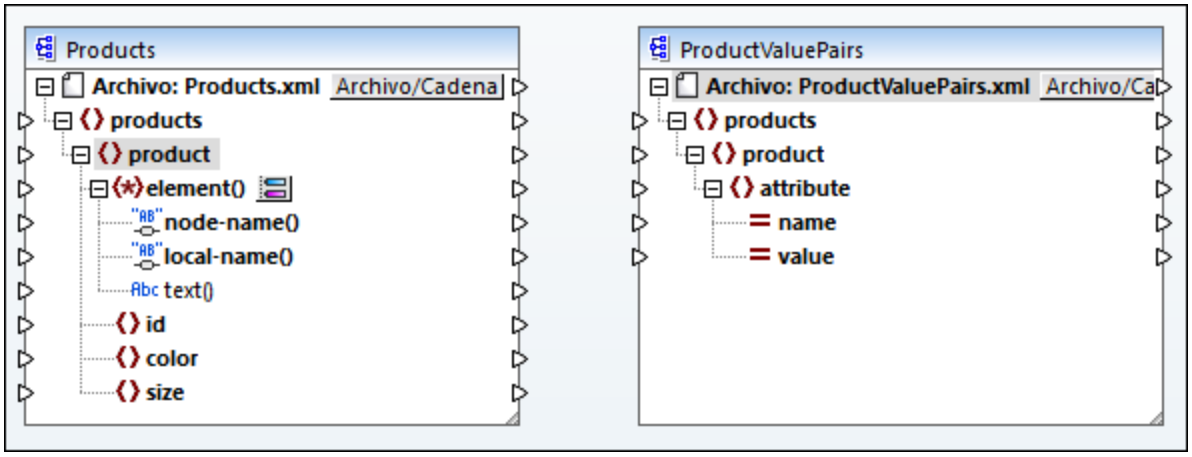


Paso nº3: habilitar el acceso dinámico a los nodos secundarios

1. Haga clic con el botón derecho en el nodo `product` del componente de origen y seleccione **Mostrar elementos secundarios con nombre dinámico** en el menú contextual.
2. En el cuadro de diálogo que aparece seleccione el tipo **text()** y deje las demás opciones como están.

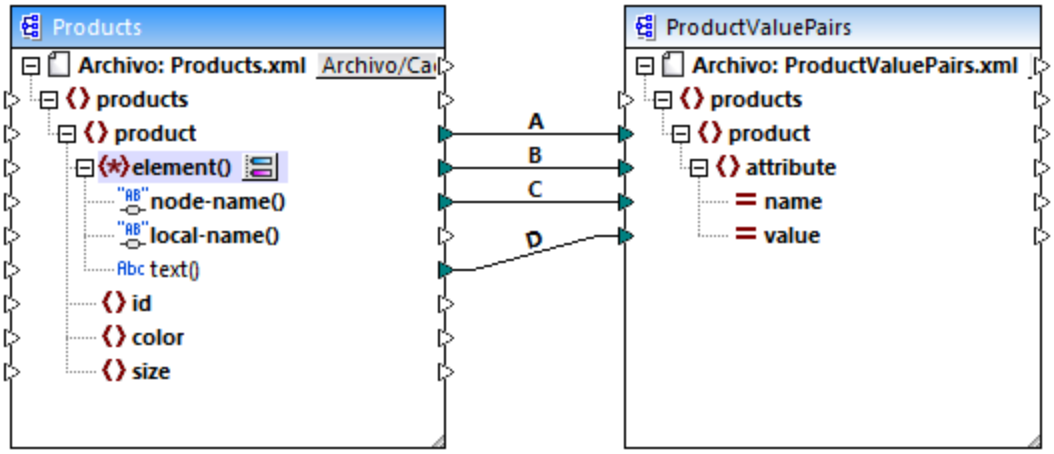


Observe que esto añade un nodo llamado `text()` al componente de origen. Este nodo se encargará de suministrar el contenido de los elementos secundarios a la asignación (en este caso, el valor de "id", "color" y "size").



Paso nº4: dibujar las conexiones de asignación de datos

Por último, debemos dibujar las conexiones de asignación de datos A, B, C y D que pueden verse en la imagen siguiente (si quiere, haga doble clic en cada línea de conexión, empezando por la primera, e introduzca el texto "A", "B", "C" y "D" respectivamente en el cuadro *Descripción*.



ConvertProducts.mfd

En el diseño de asignación anterior, la conexión A crea en el componente de destino un producto por cada producto del componente de origen. Se trata de una conexión estándar de MapForce que no se ocupa de los nombres de los nodos. Sin embargo, la conexión B crea en el componente de destino un elemento nuevo llamado `attribute` por cada elemento secundario de `product` del componente de origen.

La conexión B es una conexión crucial en la asignación pues se encarga de transferir una *secuencia* de elementos secundarios de `product` desde el componente de origen al componente de destino. No transfiere los nombres ni valores propiamente dichos. Por tanto, debe entenderse de la siguiente forma: si el **element()** de origen tiene X secundarios, entonces crea X instancias de dicho elemento en el componente de destino. En este caso concreto, `product` tiene tres secundarios en el componente de

origen (`id`, `color` y `size`). Esto significa que cada `product` del componente de destino tendrá tres secundarios llamados `attribute`.

Aunque ahora no es el caso, la misma regla puede utilizarse para asignar elementos secundarios de **attribute()**: si el elemento **attribute()** de origen tiene X atributos secundarios, la conexión creará X instancias de dicho elemento en el componente de destino.

Después, la conexión C copia el nombre real de cada elemento secundario de `product` en el componente de destino (literalmente "id", "color" y "size").

Por último, la conexión D copia el valor de cada elemento secundario de `product` (como tipo `string`) en el componente de destino.

Para consultar una vista previa de los resultados de la asignación abrimos el panel *Resultados* y consultamos el XML que se genera. Tal y como esperábamos, el resultado contiene varios productos cuyos datos están almacenados como pares nombre/valor.

```
<?xml version="1.0" encoding="UTF-8"?>
<products xsi:noNamespaceSchemaLocation="ProductValuePairs.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <product>
      <attribute name="id" value="1"/>
      <attribute name="color" value="red"/>
      <attribute name="size" value="10"/>
   </product>
   <product>
      <attribute name="id" value="2"/>
      <attribute name="color" value="blue"/>
      <attribute name="size" value="20"/>
   </product>
   <product>
      <attribute name="id" value="3"/>
      <attribute name="color" value="green"/>
      <attribute name="size" value="30"/>
   </product>
</products>
```

Resultado de la asignación

7.2 Procesar archivos por lotes

MapForce puede configurarse para procesar varios archivos simultáneamente (p. ej. todos los archivos de un directorio) cuando se ejecute la asignación. Gracias a esta característica podrá:

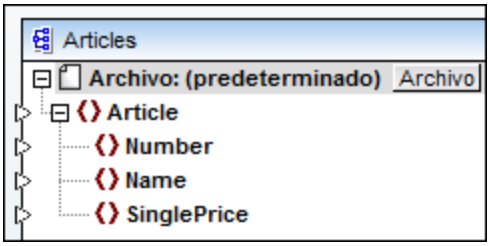
- Suministrar a la asignación una lista de archivos de entrada que se deben procesar.
- Generar como resultado de la asignación una lista de archivos en lugar de un solo archivo de salida.
- Generar una aplicación de asignación de datos donde el nombre de los archivos de entrada y salida se definen en tiempo de ejecución.
- Pasar un grupo de archivos a otro formato.
- Dividir un archivo de gran tamaño en varios archivos más pequeños.
- Agrupar varios archivos diferentes en un solo archivo.

Hay dos maneras de configurar componentes de MapForce para que procese varios archivos a la vez:

- Suministrando la ruta de acceso de los archivos de entrada/salida pertinentes por medio de caracteres de comodín (en lugar de dar un nombre de archivo fijo) en la configuración del componente (*véase* *Cambiar configuración de los componentes* 39). Es decir, puede introducir los comodines * y ? en el cuadro de diálogo "Configuración del componente" para que MapForce resuelva la ruta de acceso correspondiente cuando se ejecute la asignación.
- Conectando el nodo raíz de un componente con una secuencia que suministra la ruta de acceso de forma dinámica (p.ej. el resultado de la función `replace-fileext`). Cuando se ejecute la asignación, MapForce leerá de forma dinámica todos los archivos de entrada o generará todos los archivos de salida de forma dinámica.

Dependiendo de cual sea el objetivo del proyecto, podrá usar una de estas técnicas o ambas en la misma asignación. Sin embargo, no tiene sentido utilizar ambas técnicas al mismo tiempo en el mismo componente.

Para indicar qué técnica desea usar para cada componente haga clic en el botón Archivo o Archivo/Cadena disponible junto al nodo raíz de cada componente. Estos son los comportamientos que puede especificar con estos botones:

Usar nombres de archivo de la configuración del componente	Si el componente debe procesar un archivo de instancia o varios, esta opción ordena el procesamiento de los nombres de archivo definidos en el cuadro de diálogo "Configuración del componente". Si selecciona esta opción, el nodo raíz no tiene un conector de entrada porque no es relevante. 

	Si todavía no ha especificado archivos de entrada/salida en el cuadro de diálogo "Configuración del componente", el nombre del nodo raíz será **`Archivo: (predeterminado)`**. De lo contrario el nodo raíz muestra el nombre del archivo de entrada seguido de un punto y coma más el nombre del archivo de salida. Si el nombre de la entrada es idéntico al archivo de salida, entonces aparece como nombre del nodo raíz.  Esta opción y la opción **Usar nombres de archivo dinámicos dados por la asignación** se excluyen mutuamente.
Usar nombres de archivo dinámicos dados por la asignación	Esta opción ordena el procesamiento de los nombres de archivo definidos en el área de asignación al conectar valores con el nodo raíz del componente. Si selecciona esta opción, el nodo raíz recibe un conector de entrada con el que se pueden conectar valores que suministren de forma dinámica los nombres de archivo que se deben procesar durante la ejecución de la asignación. Si también definió nombres de archivo en el cuadro de diálogo "Configuración del componente", dichos valores se omitirán. Cuando se selecciona esta opción, el nombre del nodo raíz aparece como **`Archivo: <dinámico>`**.  Esta opción y la opción **Usar nombres de archivo de la configuración del componente** se excluyen mutuamente.

Estos son los componentes donde se pueden definir varios archivos de entrada/salida:

- Archivos XML
- Archivos de texto (archivos CSV*, FLF* y FlexText**)
- Documentos EDI**
- Hojas de cálculo Excel**
- Documentos XBRL**
- Archivos JSON**
- Archivos Protocol Buffers**

** Con MapForce Professional Edition*
*** Con MapForce Enterprise Edition*

En esta tabla puede ver la compatibilidad de cada lenguaje de MapForce con archivos de entrada/salida dinámicos y con comodines:

Lenguaje de destino	Nombre de archivo de entrada dinámico	Nombre de archivo de entrada admite comodines	Nombre de archivo de salida dinámico
XSLT 1.0	*	Incompatible con XSLT 1.0	Incompatible con XSLT 1.0
XSLT 2.0	*	*(1)	*
XSLT 3.0	*	*(1)	*
C++	*	*	*
C#	*	*	*
Java	*	*	*
BUILT-IN (motor integrado)	*	*	*

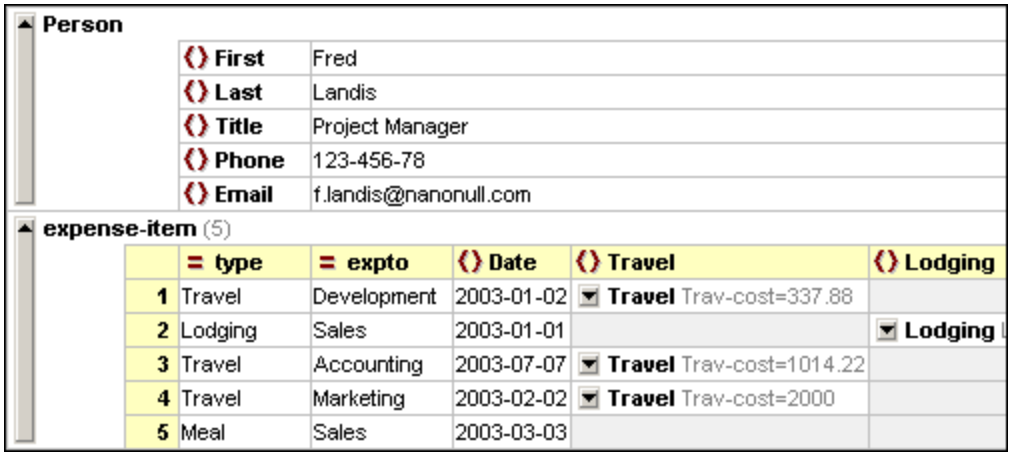
Leyenda:

*	Compatible
(1)	XSLT 2.0, XSLT 3.0 y XQuery utilizan la función `fn:collection`. La implementación en los motores XSLT 2.0, XSLT 3.0 y XQuery de Altova resuelve comodines. Los demás motores pueden comportarse de modo distinto.

7.2.1 Ejemplo: dividir un archivo XML en varios archivos

Este ejemplo demuestra cómo se pueden generar varios archivos XML de forma dinámica a partir de un solo archivo XML de origen. Este ejemplo es el archivo de muestra **<Documentos>\Altova\MapForce2024\MapForceExamples\Tutorial\Tut-ExpReport-dyn.mfd**.

El archivo XML de origen (situado en la misma carpeta que la asignación) está compuesto por un informe de gastos de alguien llamado *Fred Landis* y contiene cinco conceptos de gastos de diferentes tipos. El objetivo de esta asignación de datos es generar un archivo XML por cada concepto de gasto.



Person

First	Fred
Last	Landis
Title	Project Manager
Phone	123-456-78
Email	f.landis@nanonull.com

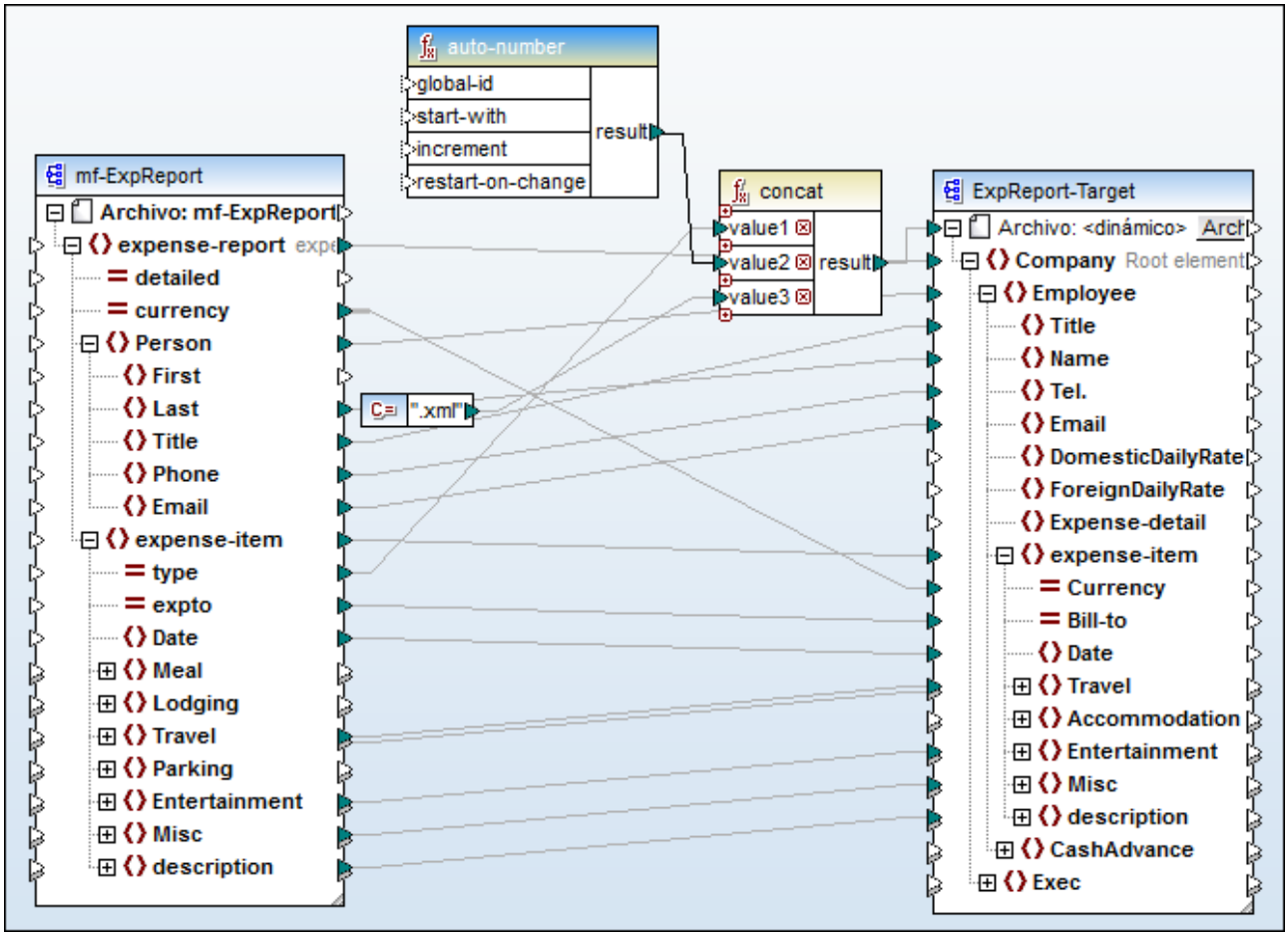
expense-item (5)

	type	expto	Date	Travel	Lodging
1	Travel	Development	2003-01-02	**Travel** Trav-cost=337.88	
2	Lodging	Sales	2003-01-01		**Lodging**
3	Travel	Accounting	2003-07-07	**Travel** Trav-cost=1014.22	
4	Travel	Marketing	2003-02-02	**Travel** Trav-cost=2000	
5	Meal	Sales	2003-03-03		

mf-ExpReport.xml (en la vista Cuadrícula de Altova XMLSpy)

Como el atributo `type` define el tipo de gasto, este será el elemento que usaremos para dividir el archivo de origen. Para conseguir nuestro objetivo debemos seguir estos pasos:

1. Insertamos una función **`concat`** (se puede arrastrar desde la biblioteca de funciones **core | string functions** de la ventana Bibliotecas).
2. Insertamos una constante (clic en el comando **Insertar | Constante**) e introducimos el valor ".xml".
3. Insertamos la función **`auto-number`** (se puede arrastrar desde la biblioteca de funciones **core | generator functions** de la ventana Bibliotecas).
4. Hacemos clic en el botón Archivo o Archivo/Cadena del componente de destino y seleccionamos **Usar nombres de archivo dinámicos dados por la asignación**.
5. Para terminar creamos las conexiones que se ven en la imagen siguiente y hacemos clic en el panel *Resultados* para ver el resultado de la asignación.



Tut-ExpReport-dyn.mfd (MapForce Basic Edition)

Tenga en cuenta que los archivos de salida resultantes tendrán nombres dinámicos construidos de la siguiente manera:

- El atributo `type` aporta la primera parte del nombre de archivo (p. ej. "Travel").
- La función **`auto-number`** aporta el número secuencial del archivo (p. ej. "Travel1", "Travel2" y así sucesivamente).
- La constante aporta la extensión del archivo, que es ".xml". Por tanto, el nombre del primer archivo de salida es "Travel1.xml".

7.3 Reglas y estrategias de asignación

Por lo general, MapForce asigna los datos de manera intuitiva, pero puede que encuentre situaciones en las que los resultados contengan demasiados o demasiados pocos elementos. En este apartado le explicaremos cómo evitar situaciones en las que la asignación genera un resultado no deseado debido a conexiones o un contexto de asignación incorrectos.

Reglas de asignación

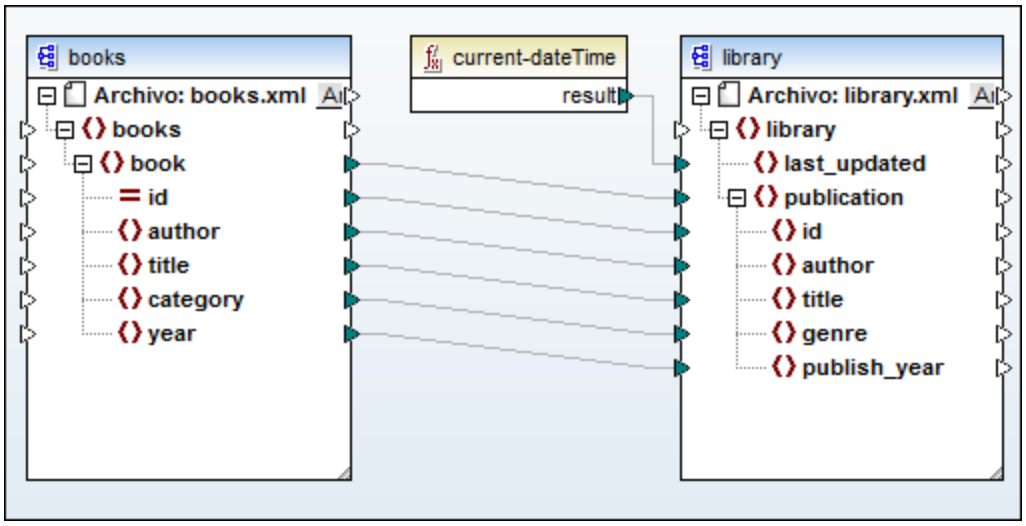
Para que una asignación sea válida, esta debe incluir al menos un componente de entrada y uno de salida. Un componente de entrada es aquel que lee datos, normalmente de un archivo o de una base de datos. Un componente de destino es el que escribe datos, normalmente en un archivo o en una base de datos. Si intenta guardar una asignación que no cumpla estos requisitos aparece un error en la ventana Mensajes: "Una asignación necesita tener al menos dos estructuras conectadas, por ejemplo un esquema o una estructura de BD".

Para crear una asignación de datos debe trazar las conexiones entre los elementos de los componentes de entrada y de salida.

Todas las conexiones de asignación que trace conforman el algoritmo de asignación. En el momento de la ejecución, MapForce evalúa el algoritmo y procesa los datos basándose en él. La integridad y la eficiencia del algoritmo de asignación depende principalmente de estas conexiones. También puede adaptar algunas opciones a nivel de asignación 76, de componente 39 o incluso de conexión 50, pero básicamente son las conexiones de la asignación las que determinan cómo se procesan los datos.

Tenga en cuenta estas reglas al crear las conexiones:

1. Al trazar una conexión *desde* un elemento de entrada, la asignación lee datos asociados a él en el archivo o la BD de entrada. Los datos pueden aparecer ninguna vez, una o varias veces (es decir, pueden ser una secuencia, como se describe a continuación). Por ejemplo, si la asignación lee datos de un archivo XML que contiene libros, el archivo XML de entrada puede contener ninguno, uno o varios elementos **book**. En la asignación siguiente, observe que el elemento **book** aparece solamente una vez en el componente de la asignación, aunque puede que el archivo (de instancia) de origen contenga varios elementos **book** o ninguno.



2. Al trazar una conexión a un elemento de destino, la asignación genera datos de instancia de esa clase. Si el elemento de entrada contiene contenido simple (por ejemplo, una cadena o un número entero) y el elemento de destino acepta contenido simple, entonces MapForce copia el contenido al elemento de destino y, si lo necesita, convierte el tipo de datos. Se pueden generar ningún valor, uno o varios en función de los datos de entrada (*véase el punto siguiente*).
3. Para cada elemento (de instancia) en el origen se crea un elemento (de instancia) en el destino. **Esta es la regla general de asignación en MapForce**. Si tomamos como ejemplo la asignación anterior, si el XML de origen contiene tres elementos **book**, entonces se crean tres elementos **publication** en el lado de destino. Tenga en cuenta que también existen algunos casos especiales (*véase el apartado* *Secuencias* 408).
4. Cada conexión crea un *contexto actual de asignación*. El contexto determina qué datos están disponibles *en el momento actual* para el *nodo de destino actual*. Por lo tanto, el contexto determina qué elementos de origen se copian de en el componente de destino. Al trazar u omitir una conexión puede cambiar el contexto actual sin querer, lo que afectaría al resultado de la asignación. Por ejemplo, puede que la asignación en la que está trabajando haga varias llamadas innecesarias a una BD o a un servicio web dentro de la misma ejecución de la asignación. Este concepto se describe con más detalle en Contexto de la asignación 409.

7.3.1 Secuencias

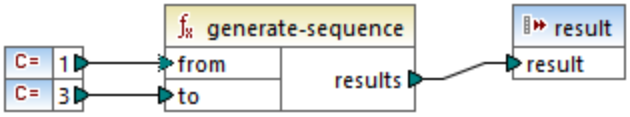
Como hemos mencionado anteriormente, la regla general de asignación es "por cada elemento de origen se crea uno de destino". Aquí, "elemento" puede significar:

- Una única instancia del archivo o la BD de entrada
- Una secuencia de cero a varios nodos de instancia del archivo o la BD de entrada

Durante la ejecución de la asignación, si una secuencia alcanza un elemento de destino se crea un bucle que genera tantos nodos de destino como nodos de origen haya. Sin embargo, hay algunas excepciones a esta regla:

- Si el elemento de destino es un elemento XML raíz, entonces sólo se crea una vez. Si conecta a él una secuencia, el resultado puede no ser un esquema válido. Si también se conectan a él atributos del elemento raíz, entonces la serialización XML fallará en tiempo de ejecución. Esto quiere decir que debe evitar conectar secuencias a elementos XML raíz.

- Si el elemento de destino acepta solamente un valor, se crea una sola vez. Algunos ejemplos de elementos que aceptan solamente un valor: atributos XML, campos de BD, componentes simples de salida. Por ejemplo, la asignación siguiente genera una secuencia de tres números enteros (1, 2, 3) con ayuda de la función **generate-sequence**. Sin embargo, el resultado contendrá solamente un número entero porque el destino es un componente simple de destino que acepta solamente un valor. Los otros dos valores se ignoran.



- Si el esquema de origen indica que un elemento específico ocurre una sola vez pero el archivo de instancia contiene varios, MapForce extrae el primer elemento del origen (que, conforme al esquema, debe existir) y crea un solo elemento en el componente de destino. Para deshabilitar este comportamiento, desmarque la casilla *Optimi&zación de procesamiento de datos de entrada basada en minOccurs/maxOccurs* en la configuración del componente (*véase también* *Configuración de componentes XML* 114).

Si la secuencia está vacía no se genera nada en el lado de destino. Por ejemplo, si el componente de destino es un documento XML y la secuencia de origen está vacía, entonces no se crearía ningún elemento XML en el componente de destino.

Las funciones funcionan de forma parecida: si obtienen una secuencia como entrada, entonces se les llama tantas veces como (y generan tantos resultados como) elementos haya en la secuencia.

> Si una función obtiene una secuencia vacía como entrada, también devuelve un resultado vacío, por lo que no genera ninguna salida.

Sin embargo, hay algunas categorías o funciones que, por motivos de diseño, devuelven un valor incluso aunque obtengan una secuencia vacía como entrada:

- **exists**, **not-exists**, **substitute-missing**
- **is-null**, **is-not-null**, **substitute-null** (estas tres funciones son alias de las tres anteriores)
- funciones agregadas (**sum**, **count**, etc.)
- funciones definidas por el usuario que aceptan secuencias y son funciones regulares (no inline)

Si necesita reemplazar algún valor vacío, añada la función **substitute-missing** a la asignación y reemplace el valor vacío con otro valor.

Las funciones pueden tener distintos elementos de entrada. Si se conecta una secuencia a cada elemento de entrada, esto genera un producto cartesiano de todos los elementos de entrada, que no suele ser el resultado deseado..Para evitarlo, conecte solamente una secuencia a una función con varios parámetros: todos los demás parámetros se deben conectar a elementos "únicos" de elementos de nivel superior u otros componentes.

7.3.2 Contexto de la asignación

Los elementos de una asignación son estructuras jerárquicas que pueden contener muchos niveles de profundidad. Por otro lado, una asignación puede tener varios componentes de origen y de destino, además de

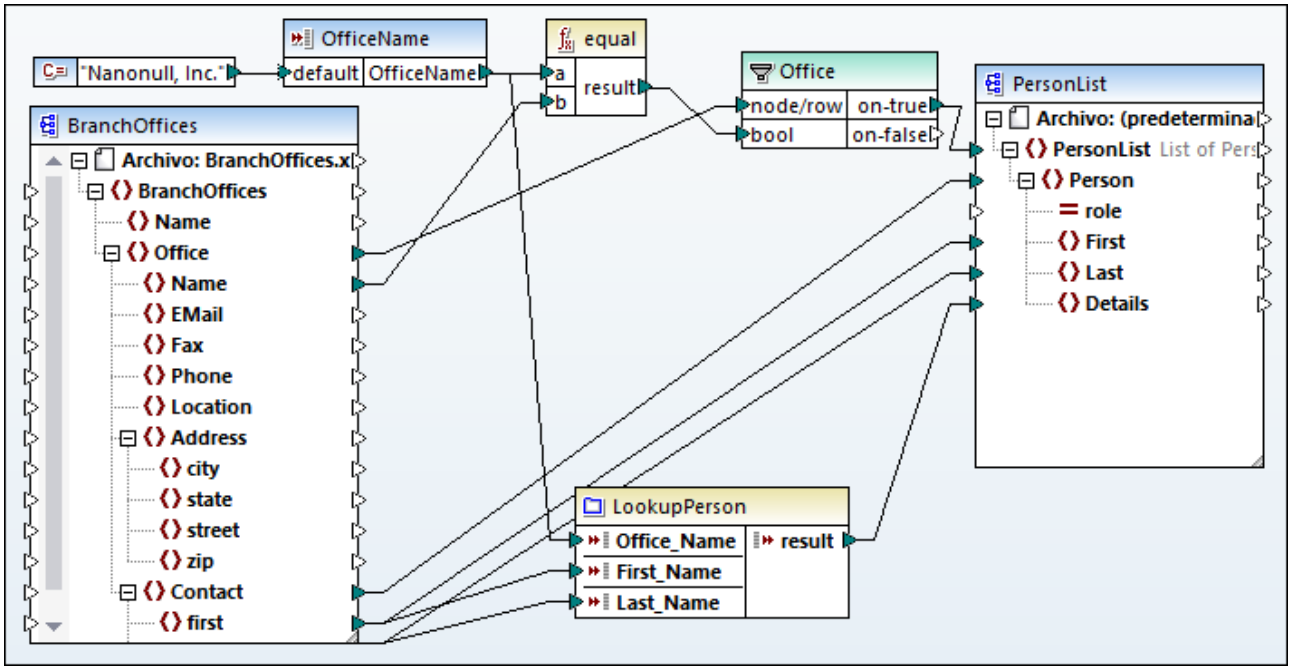
otros componentes intermedios como funciones, filtros, pares valor-asignación, etc. Esto añade complejidad al algoritmo de la asignación, especialmente si se conectan varios componentes no relacionados. Para poder ejecutar una asignación por partes, es decir, un paso cada vez, se debe estableces un contexto actual para cada conexión.

También se podría decir que se establecen varios "contextos actuales" para la duración de la ejecución de la asignación, ya que el contexto actual va cambiando según se van procesando las conexiones.

MapForce siempre establece el contexto actual empezando por el (*nodo*) *elemento raíz de destino*. Ese es el punto en el que empieza la ejecución de la asignación. La conexión con el elemento raíz de destino se traza de vuelta a todos los elementos de origen que están directa o indirectamente conectados a él, también mediante funciones u otros componentes intermedios. Todos los elementos de origen y los resultados producidos por las funciones se añaden al contexto actual.

Una vez se ha terminado de procesar el nodo de destino, MapForce empieza a avanzar por la estructura jerárquica. Más concretamente, procesa todos los *elementos asignados* del componente de destino de arriba a abajo. Por cada elemento nuevo se establece un contexto nuevo que inicialmente contiene todos los elementos del contexto matriz.Por lo tanto, todos los elementos asignados del mismo nivel que haya en un componente de destino son independientes unos de otros pero tienen acceso a todos los datos de origen de los elementos correspondientes de nivel superior.

Ahora vamos a ver cómo funciona en la práctica lo que acabamos de explicar; para ello usaremos la asignación de ejemplo **PersonListByBranchOffice.mfd**, que puede encontrar en el directorio **<Documentos>\Altova\MapForce2024\MapForceExamples**.



En la asignación anterior, tanto el componente de origen como el de destino son archivos XML. El archivo XML de origen contiene dos elementos **Office**.

Como hemos mencionado previamente, la ejecución de la asignación siempre empieza por el nodo raíz de destino (**PersonList** en este ejemplo). Al trazar de vuelta la conexión (con el filtro y la función) a un elemento

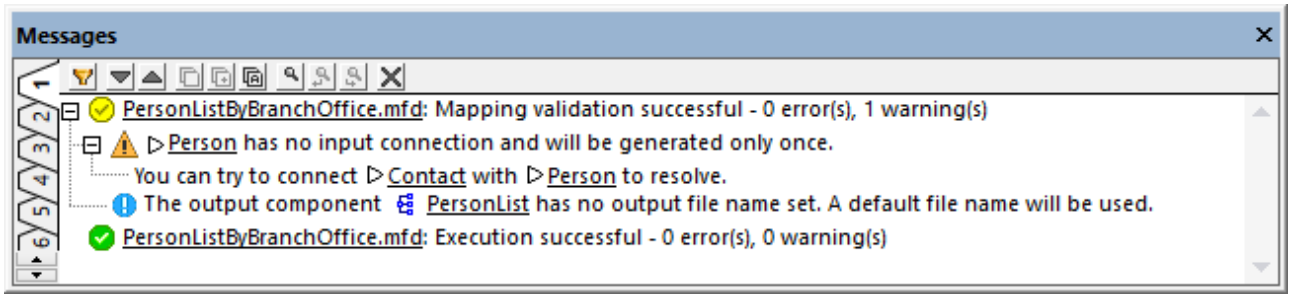
de origen, se puede concluir que el elemento de origen es **Office**. (La otra ruta de conexión lleva a un parámetro de entrada cuyo propósito se explica más abajo).

Si hubiera habido una conexión directa entre **Office** y **PersonList**, entonces, conforme a la regla general de asignación, la asignación habría creado tantos elementos de instancia `PersonList` como elementos **Office** hay en el archivo de origen. Sin embargo, esto no es lo que ocurre aquí porque hay un filtro entre medias. El filtro sólo pasa al componente de destino datos que cumplen con la condición booleana que hay conectada al elemento de entrada `bool` del filtro. La función `equal` devuelve `true` si el nombre de la oficina es igual a "Nanonull, Inc.". Esta condición se cumple una sola vez porque sólo existe un nombre así en el archivo XML de origen.

En consecuencia, la conexión entre **Office** y **PersonList** define una única oficina como el contexto para todo el documento de destino. Esto significa que todos los elementos secundarios de **Personlist** tienen acceso a los datos de la oficina "Nanonull, Inc." y que no existe ninguna otra oficina en el contexto actual.

La siguiente conexión es entre **Contact** y **Person**. Según la regla general de asignación se crea un elemento **Person** de destino por cada elemento **Contact** de origen. En cada iteración esta conexión establece un contexto actual nuevo; por lo tanto, las conexiones secundarias (**first** a **First**, **last** a **Last**) suministran datos del elemento de origen al de destino en el contexto de cada elemento **Person**.

Si omite la conexión entre **Contact** y **Person**, entonces la asignación crearía solamente un elemento **Person** con varios nodos **First**, **Last** y **Details**. En estos casos, MapForce emite un mensaje de advertencia y una sugerencia en la ventana Mensajes, por ejemplo:



Finalmente, la asignación incluye una función definida por el usuario, `LookupPerson`. La función definida por el usuario también se ejecuta en el contexto de cada elemento **Person** debido a la conexión primaria entre **Contact** y **Person**. Más concretamente, cada vez que un elemento **Person** nuevo se crea en el lado de destino, se llama a la función para que rellene el elemento **Details** de esa persona. Esta función toma tres parámetros de entrada. El primero (**OfficeName**) está definido para que lea datos del parámetro de entrada de la asignación. Los datos de origen para este parámetro también los podría suministrar el elemento de origen **Name** sin que eso cambiara el resultado de la asignación. En ambos casos el valor de origen es el mismo y se toma de un contexto de nivel superior. A nivel interno, la función de búsqueda concatena los valores recibidos como argumentos y produce un valor único. Para más información sobre cómo funciona la función `LookupPerson` consulte Ejemplo: búsqueda y concatenación 219.

7.3.2.1 Funciones definidas por el usuario

Las funciones definidas por el usuario son funciones personalizadas incrustadas en la asignación, donde se definen los componente de entrada y de salida, así como la lógica de procesamiento. Cada función definida por el usuario puede contener los mismos tipos de componentes que una asignación principal, incluidos servicios web y bases de datos.

Por defecto, si una función definida por el usuario contiene una BD o un componente de servicio web y si los datos de entrada a esa función son una secuencia de varios valores, entonces cada valor de entrada llamará a la función, lo que resultará en una llamada a la BD o al servicio web.

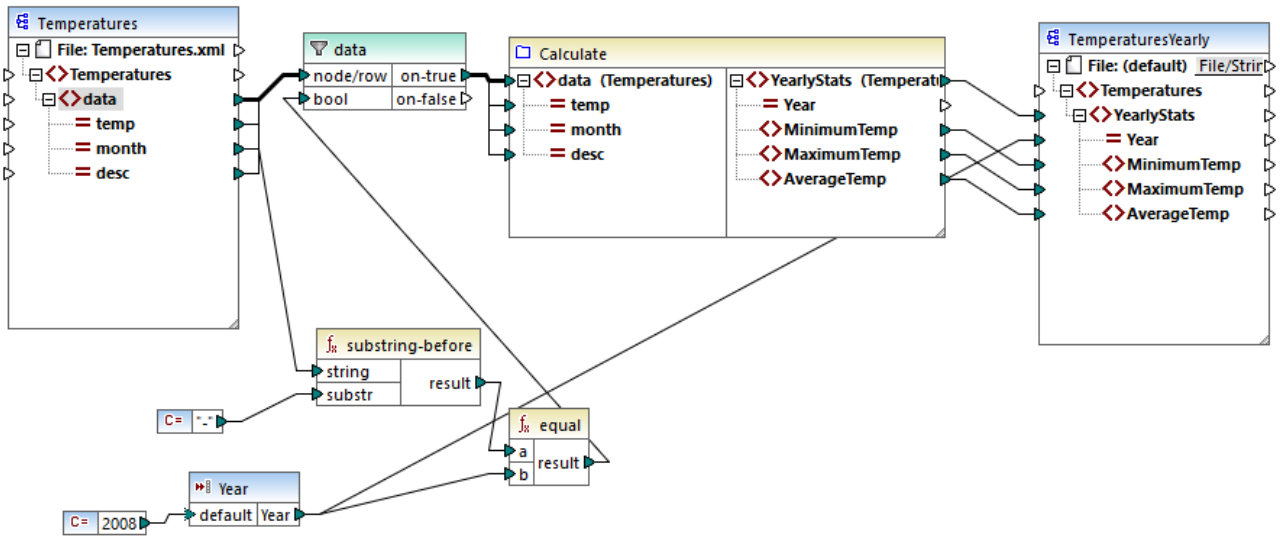
Este comportamiento puede ser aceptable para asignaciones en las que realmente necesita que se llame a la función definida por el usuario *tantas veces como valores de entrada haya*, para las que no existe otra alternativa.

Si no quiere que ocurra eso, puede configurar la función definida por el usuario para que sólo se la llame una vez, incluso aunque el componente de entrada sea una secuencia de valores. Tiene sentido usar esta opción en el caso de aquellas funciones definidas por el usuario que operan en un conjunto de valores antes de devolver un resultado (como las funciones que calculan valores medios o totales).

Se pueden configurar funciones definidas por el usuario para que acepten varios valores de entrada en la misma llamada a la función si esta es de tipo "regular" y no "inline". (*Para más detalles consulte* *Funciones definidas por el usuario* 205). Para indicar en funciones regulares que el parámetro de entrada es una secuencia marque la casilla *El parámetro de entrada es una secuencia*. Esta casilla se hace visible en la configuración del componente después de hacer doble clic en la barra de título de un parámetro de entrada. La casilla afecta la cantidad de llamadas que se hacen a la función:

- Cuando los datos de entrada están conectados al parámetro **sequence** se llama *sólo una vez* a la función definida por el usuario y se le pasa la secuencia completa.
- Cuando los datos de entrada están conectados a un parámetro **non-sequence**, se llama a la función definida por el usuario *una vez por cada elemento de la secuencia*.

Para ver un ejemplo abra esta asignación: **<Documentos>\Altova\MapForce2024\MapForceExamples\\InputIsSequence.mfd**.



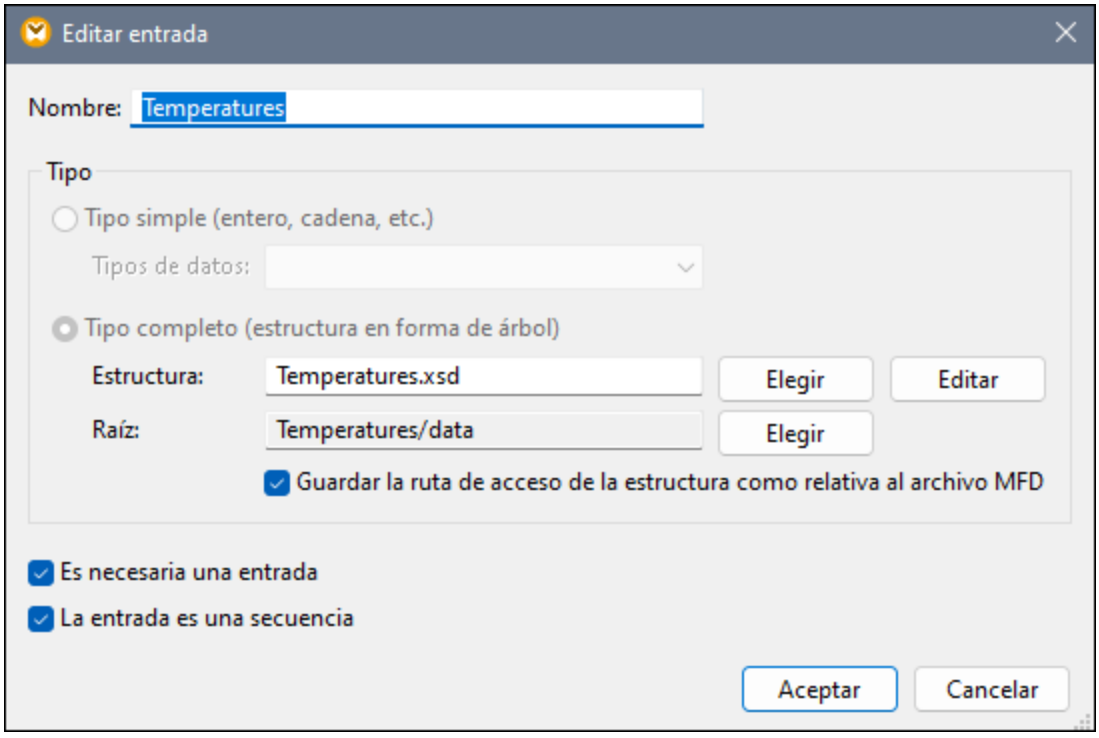
La asignación anterior ilustra un caso típico de una función definida por el usuario que opera en un conjunto de valores y, por tanto, necesita todos los valores de entrada en una sola llamada. Más concretamente, la función

definida por el usuario `Calculate` devuelve las temperaturas mínima, máxima y media, y toma sus datos de entrada de un archivo XML de origen. El resultado esperado de esta asignación es:

```
<Temperatures>
    <YearlyStats Year="2008">
        <MinimumTemp>-0.5</MinimumTemp>
        <MaximumTemp>24</MaximumTemp>
        <AverageTemp>11.6</AverageTemp>
    </YearlyStats>
</Temperatures>
```

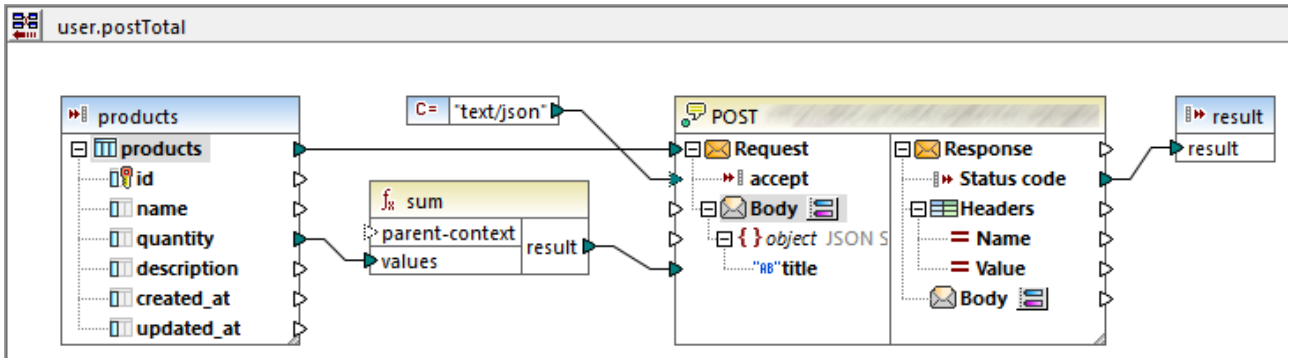
Como es habitual, la ejecución de la asignación empieza por el primer elemento del componente de destino (**YearlyStats** en este ejemplo). Para rellenar este nodo, la asignación intenta obtener datos de entrada de la función definida por el usuario, lo que a su vez desencadena el filtro. El papel del filtro en esta asignación es pasar a la función definida por el usuario solamente temperaturas del año 2008.

La casilla *El parámetro de entrada es una secuencia* se marcó para el parámetro de entrada de la función definida por el usuario (para ver esta casilla, haga clic en la barra del título de la función `Calculate` para introducir la asignación de la función y después haga doble clic en la barra del título del parámetro de entrada). Como hemos mencionado antes, marcar la casilla *El parámetro de entrada es una secuencia* hace que se suministre la secuencia completa de valores como parámetro de entrada de la función y que se llame a la función una sola vez.



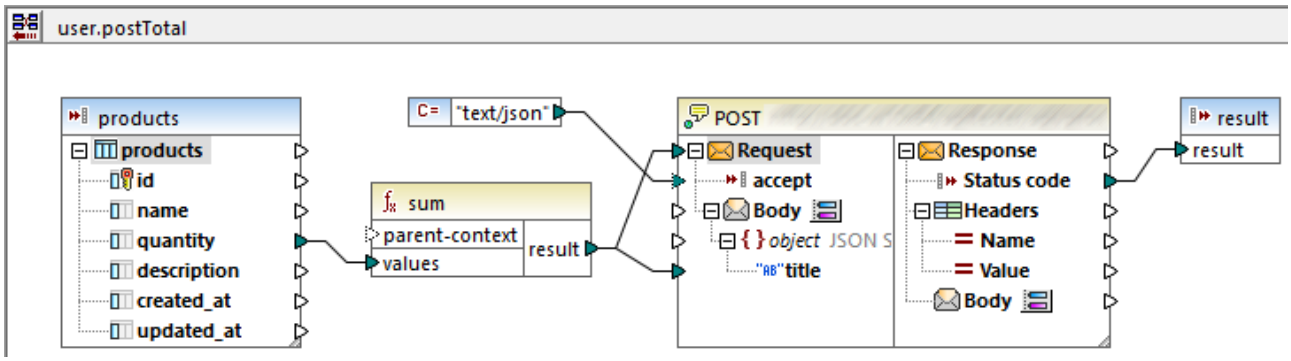
Si la casilla *El parámetro de entrada es una secuencia* no se hubiera marcado, se habría llamado a la función definida por el usuario una vez por cada valor en el componente de origen. Como resultado, se habría calculado el valor máximo, mínimo y medio para cada valor individualmente y se habría producido un resultado incorrecto.

Al aplicar la misma lógica en funciones definidas por el usuario más complejas que incluyen llamadas a BD o a servicios web, es posible optimizar la ejecución y evitar llamadas innecesarias a la BD o al servicio web. Sin embargo, tenga en cuenta que la casilla *El parámetro de entrada es una secuencia* no controla qué le ocurre a la secuencia de valores *después* de que entre en la función. En otras palabras, nada le impide conectar la secuencia de valores de entrada al componente de entrada de un servicio web y llamarlo varias veces. Considere este ejemplo:



La función definida por el usuario de la imagen anterior recibe una secuencia de valores de la asignación externa. Más concretamente, los datos suministrados al parámetro de origen provienen de una BD. Se ha marcado la casilla *El parámetro de entrada es una secuencia* para el parámetro de origen, por lo que se suministra a la función toda la secuencia en una sola llamada. La función debe sumar varios valores **quantity** y enviar el resultado en un servicio web. Se espera exactamente una llamada al servicio web. Sin embargo, cuando se ejecute la asignación se harán varias llamadas al servicio web. El motivo es que el componente de entrada **Request** del servicio web recibe una *secuencia de valores* y no un único valor.

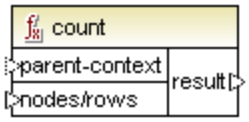
Para solucionar este problema, conecte el componente de entrada **Request** del servicio web al resultado de la función `sum`. La función produce un solo valor, por lo que también se llama al servicio web una sola vez:



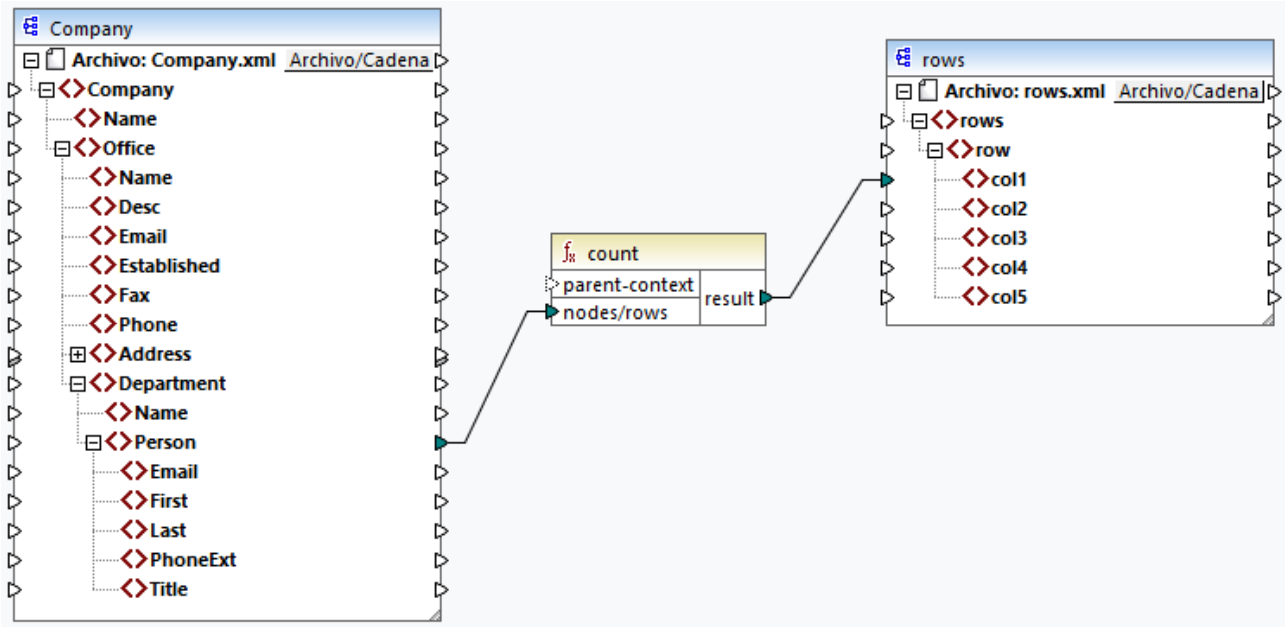
Normalmente, las funciones agregadas como **`sum`**, **`count`**, etc. producen un único valor. Sin embargo, si hay una conexión de nivel superior que lo permita, puede que también produzcan una secuencia de valores, como se describe en más detalle en Ejemplo: cambiar el contexto matriz 415.

7.3.2.2 Ejemplo: cambiar el contexto matriz

Algunos componentes de asignación tienen un elemento **parent-context** opcional. Con ayuda de este elemento puede influir en el contexto de la asignación en el que debe operar ese componente y, por tanto, cambiar el resultado de la asignación. Los componentes que tienen un elemento **parent-context** opcional son: funciones agregadas, variables y componentes `Join`.



Para ver un ejemplo de la utilidad de cambiar el contexto matriz, abra esta asignación: **<Documentos>\Altova\MapForce2024\MapForceExamples\Tutorial\ParentContext.mfd**.



En el XML de origen de la asignación anterior existe un único nodo Company que contiene dos nodos **Office**. Cada nodo **Office** contiene varios nodos **Department**, cada uno de los cuales contiene varios nodos **Person**. Si abre el archivo XML en un editor XML puede ver que la distribución de personas por oficina y departamento es esta:

Oficina	Departmento	Número de personas
Nanonull, Inc.	Administración	3
	Marketing	2
	Ingeniería	6
	IT & Soporte técnico	4
Nanonull Partners, Inc.	Administración	2

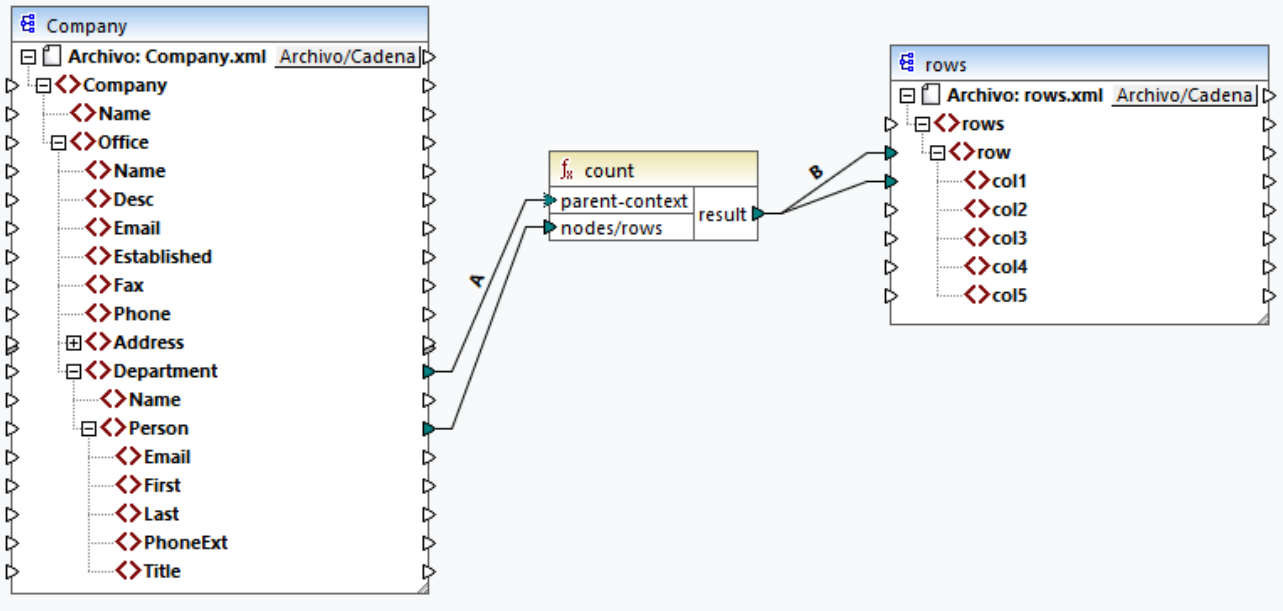
Oficina	Departmento	Número de personas
	Marketing	1
	IT & Soporte técnico	3

La asignación cuenta todas las personas de todos los departamentos. Para ello usa la función `count` de la biblioteca `core`. Si hace clic en la pestaña Resultados para previsualizar la asignación verá que produce un único valor, **21**, que corresponde al número total de personas del archivo XML.

La asignación funciona así:

- Como es habitual, la asignación empieza por el nodo de nivel superior del componente de destino (**rows** en este ejemplo). No hay ninguna conexión entrante para **rows**. Como resultado, el contexto implícito de asignación está establecido entre **Company** (elemento de nivel superior del componente de origen) y **rows** (elemento superior del componente de destino).
- El resultado de la función es un valor único porque sólo hay un elemento **Company** en el archivo de origen.
- Para rellenar el elemento de destino **col1**, MapForce ejecuta la función `count` en el contexto matriz implícito que se menciona más arriba, por lo que contará todos los nodos **Person** de todas las oficinas de todos los departamentos.

El argumento **parent-context** de la función permite cambiar el contexto de la asignación. Esto permite, por ejemplo, contar el número de personas de cada departamento. Para ello, trace dos o más conexiones como las de la imagen siguiente:

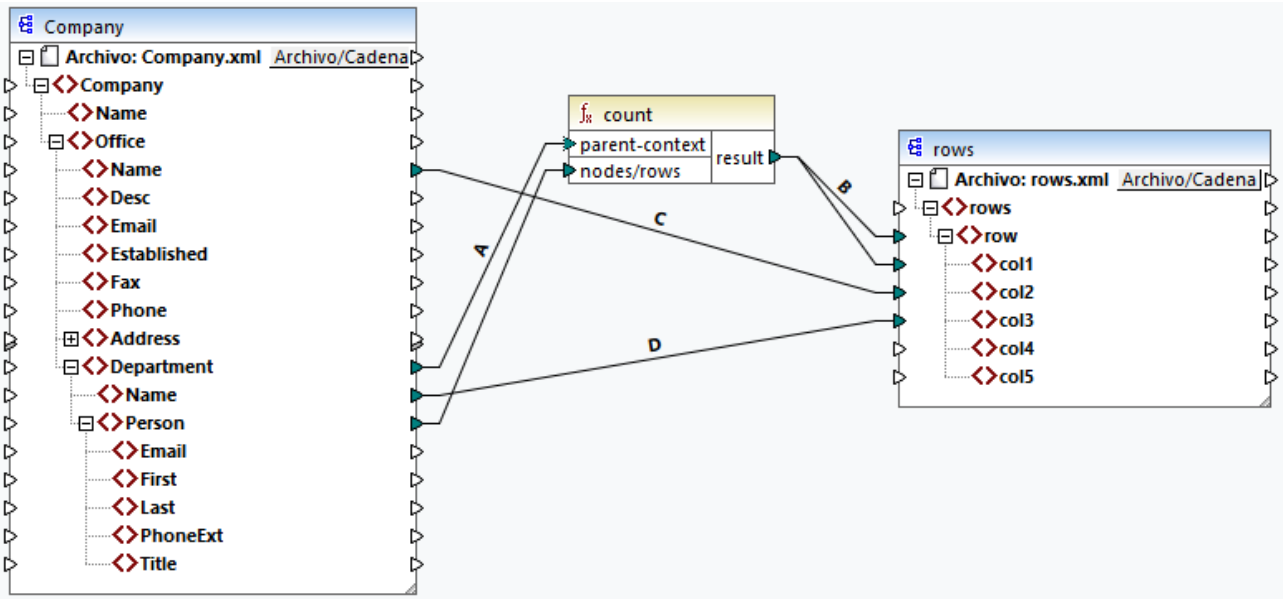


En la asignación anterior la conexión A cambia el contexto matriz de la función `count` a **Department**. Como resultado, la función contará el número de personas de cada departamento. Es importante recordar que la función ahora devolverá una *secuencia* de resultados en lugar de un solo resultado debido a que en el origen existen varios departamentos. Esta es la razón por la que existe la conexión B: por cada elemento de la secuencia resultante crea una fila nueva en el archivo de destino. El resultado de la asignación ahora cambia

en consecuencia (tenga en cuenta que los números corresponden exactamente al número de personas de cada departamento):

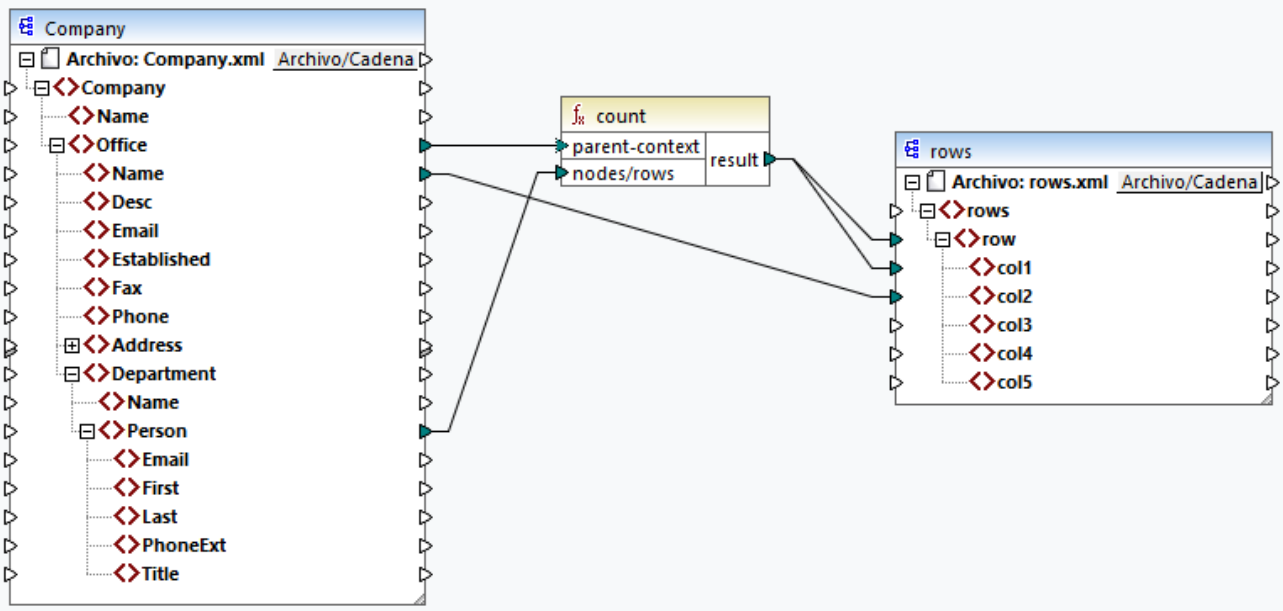
```
<rows>
    <row>
        <col1>3</col1>
    </row>
    <row>
        <col1>2</col1>
    </row>
    <row>
        <col1>6</col1>
    </row>
    <row>
        <col1>4</col1>
    </row>
    <row>
        <col1>2</col1>
    </row>
    <row>
        <col1>1</col1>
    </row>
    <row>
        <col1>3</col1>
    </row>
</rows>
```

Si tenemos en cuenta que la asignación actual crea una fila por cada departamento, puede optar por copiar también el nombre de la oficina y el del departamento en el archivo de destino; para ello debe trazar las conexiones C y D:



De este modo, el resultado no solo mostrará el número de personas, sino también los nombres de oficina y departamento correspondientes.

Si quiere contar el nombre de personas de cada oficina, conecte el contexto matriz de la función `count` al elemento **Office** del componente de destino.



Con las conexiones de la imagen anterior, la función `count` devuelve un resultado por cada oficina. En el archivo de origen existen dos oficinas, por lo que la función ahora devolverá dos secuencias. En consecuencia, en el resultado existen dos filas de las que cada una corresponde al número de personas de esa oficina.

```
<rows>
    <row>
        <col1>15</col1>
        <col2>Nanonull, Inc.</col2>
    </row>
    <row>
        <col1>6</col1>
        <col2>Nanonull Partners, Inc.</col2>
    </row>
</rows>
```

7.3.3 Contexto de prioridad

El contexto de prioridad es una forma de influir en el orden en que se evalúan los parámetros de entrada de una función. Puede que sea necesario establecer un contexto de prioridad si su asignación combina datos de dos orígenes no relacionados.

Para entender cómo funciona el contexto de prioridad, recuerde que cuando se ejecuta una asignación la conexión a un elemento de entrada puede levar una secuencia de varios valores. Para las secuencias con dos parámetros de entrada esto significa que MapForce debe crear dos bucles, uno de los cuales se debe

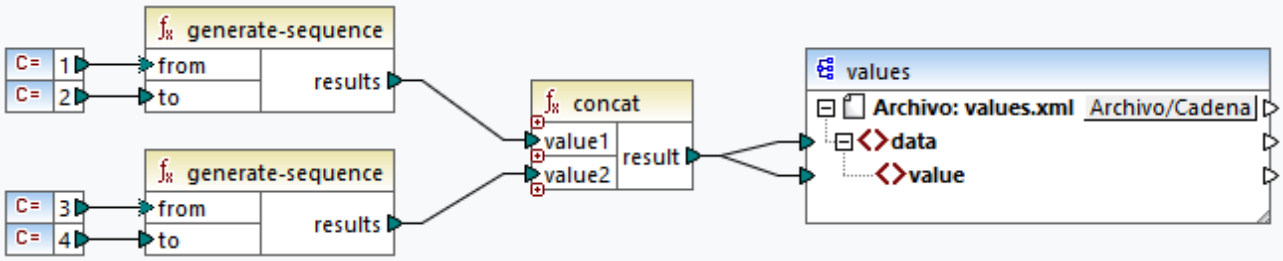
procesar primero. El bucle que se procesa primero es el bucle "exterior". Por ejemplo, la función `equal` recibe dos parámetros: *a* y *b*. Si ambos obtienen una secuencia de valores, entonces MapForce procesa así:

- Por cada ocurrencia de *a*
  - Por cada ocurrencia de *b*
    - ¿Es *a* igual a *b*?

Como puede ver en el ejemplo anterior, cada ocurrencia de *b* se evalúa en el contexto de cada ocurrencia de *a*. El contexto de prioridad permite alterar el proceso lógico para que cada ocurrencia de *a* se evalúe en el contexto de cada ocurrencia de *b*. En otras palabras, permite cambiar el bucle interno por el externo:

- Por cada ocurrencia de *b*
  - Por cada ocurrencia de *a*
    - ¿Es *a* igual a *b*?

Ahora examinemos una asignación en la que el contexto de prioridad afecta al resultado de la asignación. En la asignación siguiente la función `concat` tiene dos parámetros de entrada. Cada parámetro de entrada es una secuencia que se genera con ayuda de la función `generate-sequence`. La primera secuencia es "1,2" y la segunda es "3,4".



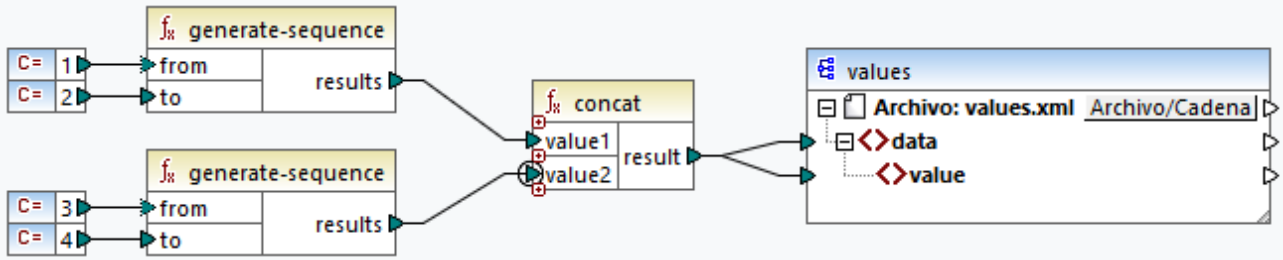
Primero, ejecutemos la asignación sin un contexto de prioridad. La función `concat` empieza por evaluar la secuencia de nivel superior, de forma que combine valores en este orden:

- 1 con 3
- 1 con 4
- 2 con 3
- 2 con 4

Esto también se refleja en el resultado de la asignación:

```
<data>
    <value>13</value>
    <value>14</value>
    <value>23</value>
    <value>24</value>
</data>
```

Si hace clic con el botón derecho en el segundo parámetro de entrada y selecciona **Contexto de prioridad** en el menú contextual, este se convierte en el contexto de prioridad, En la imagen siguiente se ve el contexto de prioridad de entrada rodeado por un círculo.



Esta vez se evalúa el segundo parámetro de entrada primero. La función `concat` sigue concatenando los mismos valores, pero esta vez procesa primero la secuencia "3,4" y el resultado cambia a:

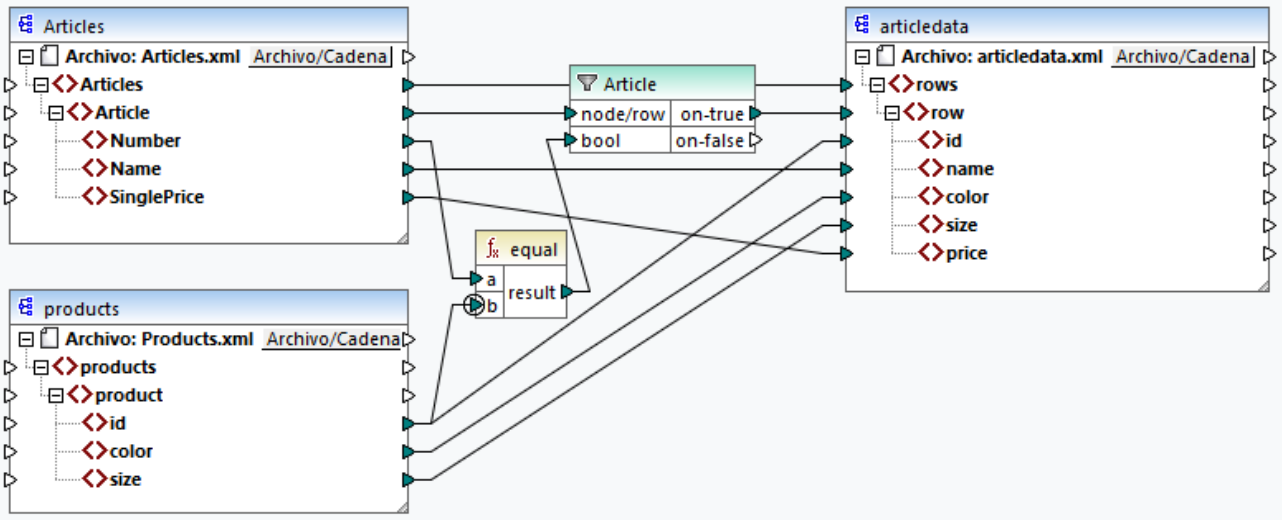
```
<data>
    <value>13</value>
    <value>23</value>
    <value>14</value>
    <value>24</value>
</data>
```

Hasta ahora sólo hemos visto la parte teórica del contexto de prioridad. Para ver un ejemplo práctico consulte Ejemplo: Filtrar con el contexto de prioridad 420.

7.3.3.1 Ejemplo: Filtrar con el contexto de prioridad

Cuando se conecta una función a un filtro, el contexto de prioridad no sólo afecta a la función en sí, sino también a la evaluación del filtro. La asignación de la imagen siguiente muestra un caso típico en el que es necesario usar el contexto de prioridad para obtener el resultado correcto. Puede encontrar esta asignación en la ruta: **<Documentos>\Altova\MapForce2024\MapForceExamples\Tutorial\FilterWithPriority.mfd**.

Nota: Esta asignación usa componentes XML, pero otros tipos de componentes de MapForce obedecen a la misma lógica, incluidos EDI, JSON, etc.



El objetivo de esta asignación es copiar datos de **Articles.xml** en un archivo XML nuevo con un esquema distinto, **articledata.xml**. Al mismo tiempo, la asignación debería buscar los detalles de cada artículo en el archivo **Products.xml** y combinarlos con el registro de archivo correspondiente. Observe que cada registro de **Articles.xml** tiene un elemento **Number** y cada registro en **Products.xml** tiene un elemento **id**. Si estos dos valores son iguales, entonces todos los otros valores () se deben copiar a la misma fila en el destino.

Para conseguir este objetivo se ha añadido un filtro. Cada filtro requiere una condición booleana como entrada; solamente aquellos nodos/filas que cumplan con esa condición se copian en el destino, para lo que se ha incluido una función `equal` a la asignación. La función `equal` comprueba si el número de artículo y el ID del producto son iguales en ambos orígenes. El resultado se suministra como entrada para el filtro. Si es **true**, entonces el elemento **Article** se copia en el destino.

Observe que se ha definido un contexto de prioridad en el segundo parámetro de entrada de la segunda función `equal`. En esta asignación no incluir el contexto de prioridad en la asignación resultaría en un resultado incorrecto.

Asignación inicial: sin contexto de prioridad

Esta es la lógica de la asignación sin contexto de prioridad:

- Conforme a la regla general de asignación, por cada elemento **Article** que cumpla con la condición del filtro se crea una fila nueva en el destino. La conexión entre Article y row (mediante la función y el filtro) se encarga de ello.
- El filtro comprueba la condición de cada artículo. Para ello, recorre todos los productos y trae varios productos al contexto actual.
- Para rellenar el **id** en el lado de destino, MapForce sigue la regla general (por cada elemento de origen se crea uno de destino). Sin embargo, como hemos explicado más arriba, todos los productos de **Products.xml** están en el contexto actual. No hay ninguna conexión entre **product** y otro lugar del destino de forma que se lea el **id** de un producto en concreto. En consecuencia, se crearán varios elementos id por cada **Article** en el destino. Lo mismo ocurre con **color** y **size**.

En resumen: los elementos de **Products.xml** tienen el contexto del filtro (que debe recorrer todos los productos); por tanto, los valores **id**, **color** y **size** se copiarán en cada fila de destino tantas veces como productos haya en el archivo de origen, lo que generará un resultado incorrecto, como este:

```
<rows>
    <row>
        <id>1</id>
        <id>2</id>
        <id>3</id>
        <name>T-Shirt</name>
        <color>red</color>
        <color>blue</color>
        <color>green</color>
        <size>10</size>
        <size>20</size>
        <size>30</size>
        <price>25</price>
    </row>
</rows>
```

Solución A: Usar el contexto de prioridad

El problema anterior se soluciona añadiendo un contexto de prioridad a la función que computa la condición booleana del filtro.

En concreto, si el segundo parámetro de entrada de la función `equal` se designa como contexto de prioridad, la secuencia proveniente de **Products.xml** se prioriza. Esto se traduce en la siguiente lógica de asignación:

- Por cada producto se rellena la entrada **b** de la función `equal` (en otras palabras, se prioriza **b**). En este punto, los detalles del producto actual están en contexto.
- Por cada artículo se rellena la entrada **a** de la función `equal` y se comprueba si la condición del filtro es **true**. Si lo es, entonces los detalles del artículo también se añaden al contexto actual.
- A continuación, se copian los detalles del artículo y del producto desde el contexto actual a los elementos correspondientes del destino.

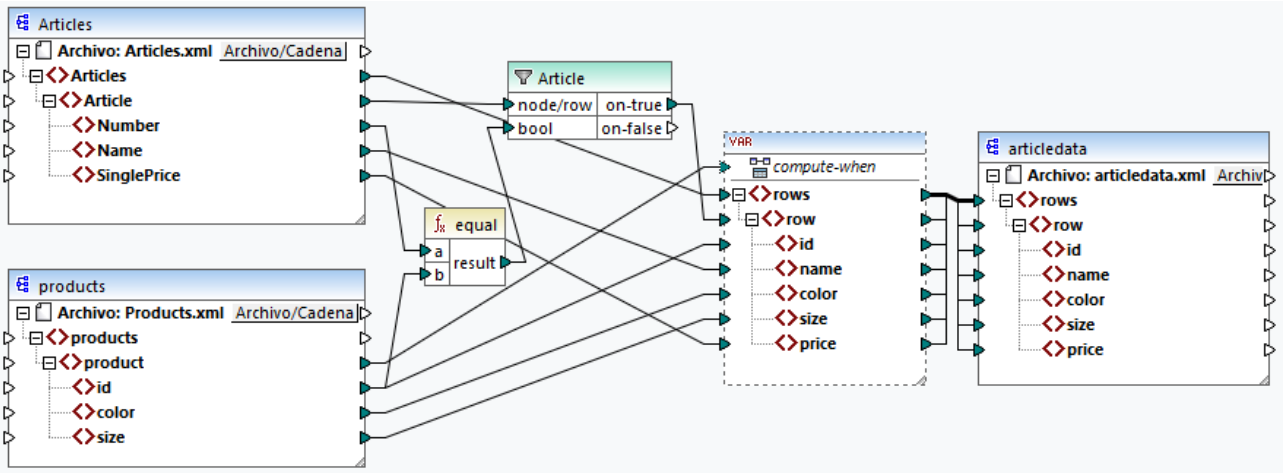
La lógica de asignación anterior produce un resultado correcto, por ejemplo:

```
<rows>
    <row>
        <id>1</id>
        <name>T-Shirt</name>
        <color>red</color>
        <size>10</size>
        <price>25</price>
    </row>
</rows>
```

Solución B: Usar una variable

Una solución alternativa es llevar al mismo contexto a cada artículo y producto que cumpla con la condición del filtro, con ayuda de una variable intermedia. Las variables se pueden usar en escenarios como este porque permiten almacenar datos de forma temporal en la asignación, lo que permite cambiar el contexto según lo necesite.

Para escenarios como este puede añadir a la asignación una variable que tiene el mismo esquema que el componente de destino. En el comando del menú **Insertar** haga clic en **Variable** y proporcione el esquema **articledata.xml** como estructura cuando la aplicación se lo pida.



En la asignación anterior ocurre lo siguiente:

- El contexto de prioridad ya no se usa. En su lugar hay una variable que tiene la misma estructura que el componente de destino.
- Como es habitual, la ejecución de la asignación empieza en el nodo raíz de destino. Antes de rellenar el destino, la asignación recopila datos en la variable.
- La variable se computa en el contexto de cada producto. Esto ocurre porque existe una conexión desde **product** hasta la entrada **compute-when** de la variable.
- La condición del filtro se comprueba en el contexto de cada producto. La estructura de la variable sólo se rellena y se pasa al destino si se cumple esta condición.

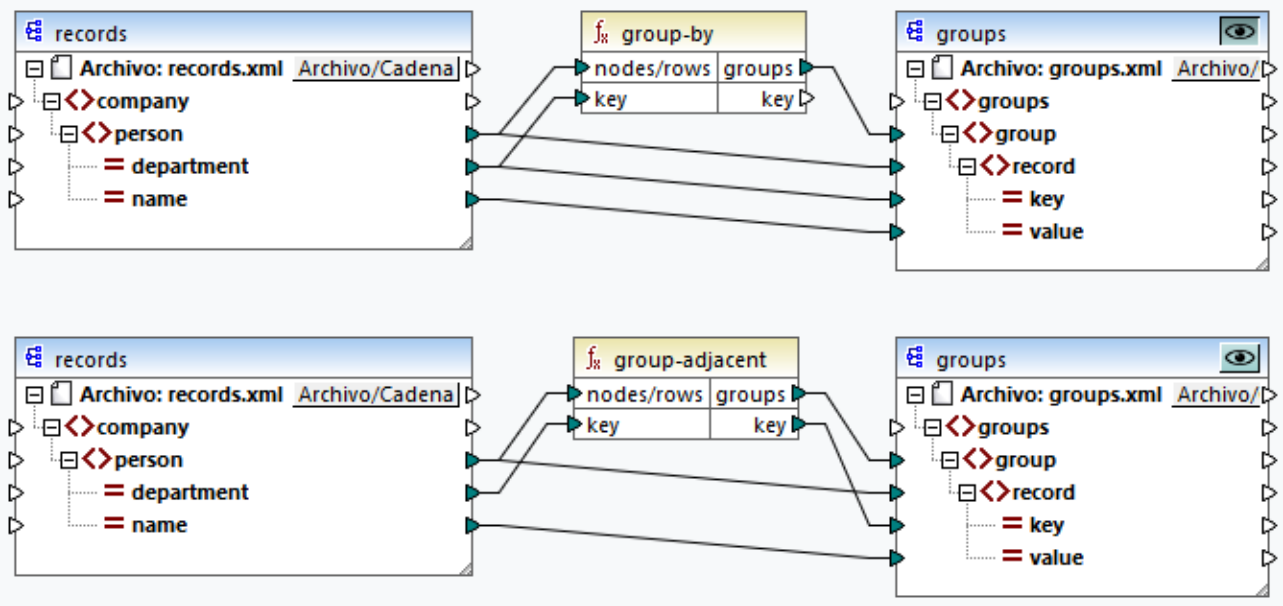
7.3.4 Varios componentes de destino

Una asignación puede tener varios componentes de origen y de destino. Cuando hay varios componentes de destino sólo puede previsualizar el resultado de un componente cada vez en MapForce, que es el que indique con el botón **Vista previa**  . En otros entornos de ejecución (MapForce Server o código generado), todos los componentes de destino se ejecutan de forma secuencial y se produce el resultado correspondiente de cada componente.

Por defecto, los componentes de destino se procesan de arriba a abajo y de izquierda a derecha. Si lo necesita, puede modificar este orden cambiando la posición de los componentes de destino en la ventana de asignación. El punto de referencia es la esquina superior izquierda de cada uno de los componentes. Tenga en cuenta lo siguiente:

- Si dos componentes tienen la misma posición vertical, entonces tiene precedencia el último.
- Si dos componentes tienen la misma posición horizontal, entonces tiene precedencia el de nivel superior.
- En el caso improbable de que varios componentes tengan la misma posición, entonces se usa automáticamente un ID de componente interno que garantiza un orden bien definido pero que no se puede modificar.

Para ver un ejemplo abra la asignación de ejemplo:
<Documentos>\Altova\MapForce2024\MapForceExamples\Tutorial\GroupingFunctions.mfd. Esta asignación consiste en varios componentes de origen y varios de destino; en la imagen se ve solamente un fragmento.



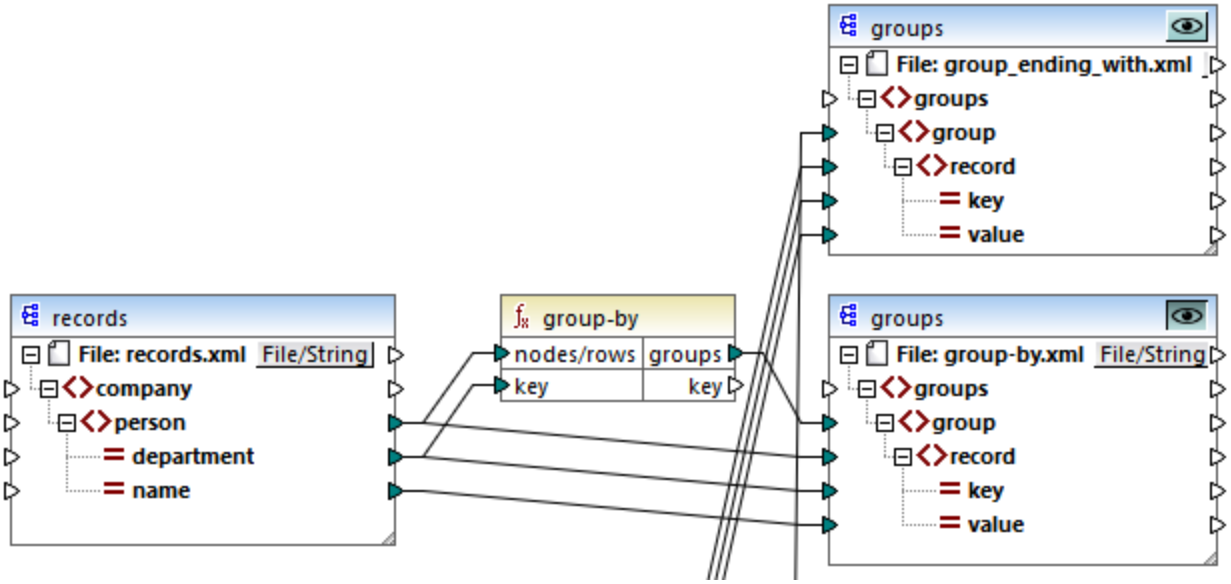
Conforme a las reglas, la orden de procesamiento predeterminada de esta asignación en MapForce Server y en código generado es de arriba a abajo. Puede comprobar que es el caso al generar código XSLT 2.0, por ejemplo:

1. En el menú **Archivo**, haga clic en **Generar código en | XSLT 2.0**.
2. Cuando la aplicación lo pida, seleccione un directorio de destino para el código generado.

Una vez lo haya generado, el directorio de destino incluye varios archivos XSLT y un archivo **DoTransform.bat**. Este último se puede ejecutar con RaptorXML Server (que requiere una licencia aparte). El archivo **DoTransform.bat** procesa componentes en el mismo orden en el que se definieron en la asignación, de arriba a abajo. Esto se puede comprobar en el parámetro `--output` de cada transformación.

```
RaptorXML xslt --xslt-version=2 --input="records.xml" --output="group-by.xml" --xml-
validation-error-as-warning=true %* "MappingMapTogroups.xslt"
IF ERRORLEVEL 1 EXIT/B %ERRORLEVEL%
RaptorXML xslt --xslt-version=2 --input="records.xml" --output="group-adjacent.xml" --
xml-validation-error-as-warning=true %* "MappingMapTogroups2.xslt"
IF ERRORLEVEL 1 EXIT/B %ERRORLEVEL%
RaptorXML xslt --xslt-version=2 --input="records.xml" --output="group-into-blocks.xml" --
xml-validation-error-as-warning=true %* "MappingMapTogroups3.xslt"
IF ERRORLEVEL 1 EXIT/B %ERRORLEVEL%
RaptorXML xslt --xslt-version=2 --input="records-v2.xml" --output="group-starting-
with.xml" --xml-validation-error-as-warning=true %* "MappingMapTogroups4.xslt"
IF ERRORLEVEL 1 EXIT/B %ERRORLEVEL%
RaptorXML xslt --xslt-version=2 --input="records-v3.xml" --output="group_ending_with.xml"
--xml-validation-error-as-warning=true %* "MappingMapTogroups5.xslt"
IF ERRORLEVEL 1 EXIT/B %ERRORLEVEL%
```

La última transformación produce un archivo de salida llamado **group-ending-with.xml**. Ahora vamos a mover este componente de destino en la asignación a la parte de más arriba:



Si ahora vuelve a generar código XSLT 2.0, el orden de procesamiento también cambia:

```
RaptorXML xslt --xslt-version=2 --input="records-v3.xml" --output="group_ending_with.xml"
--xml-validation-error-as-warning=true %* "MappingMapTogroups.xslt"
IF ERRORLEVEL 1 EXIT/B %ERRORLEVEL%
RaptorXML xslt --xslt-version=2 --input="records.xml" --output="group-by.xml" --xml-
validation-error-as-warning=true %* "MappingMapTogroups2.xslt"
IF ERRORLEVEL 1 EXIT/B %ERRORLEVEL%
RaptorXML xslt --xslt-version=2 --input="records.xml" --output="group-adjacent.xml" --
xml-validation-error-as-warning=true %* "MappingMapTogroups3.xslt"
IF ERRORLEVEL 1 EXIT/B %ERRORLEVEL%
RaptorXML xslt --xslt-version=2 --input="records.xml" --output="group-into-blocks.xml" --
xml-validation-error-as-warning=true %* "MappingMapTogroups4.xslt"
IF ERRORLEVEL 1 EXIT/B %ERRORLEVEL%
RaptorXML xslt --xslt-version=2 --input="records-v2.xml" --output="group-starting-
with.xml" --xml-validation-error-as-warning=true %* "MappingMapTogroups5.xslt"
IF ERRORLEVEL 1 EXIT/B %ERRORLEVEL%
```

En el fragmento de código anterior, la primera llamada ahora produce **group-ending-with.xml**.

Puede cambiar el orden de procesamiento de forma parecida en otros lenguajes de programación y en los archivos de ejecución compilados de MapForce Server (.mfx).

Asignaciones encadenadas

Para las asignaciones encadenadas se sigue la misma secuencia de procesamiento que se describe más arriba. Sin embargo, el grupo de asignaciones encadenadas se entiende como una unidad. Cambiar la posición del componente intermedio o el componente final de una única asignación encadenada no afecta a la secuencia de procesamiento. La posición de los componentes finales de destino de cada grupo determinan

qué se procesa primero solamente si existen varias "cadenas" o varios componentes de destino en una asignación.

- Si dos componentes de destino finales tienen la misma posición vertical, entonces tiene precedencia el último.
- Si dos componentes de destino finales tienen la misma posición horizontal, entonces tiene precedencia el de nivel superior.
- En el caso improbable de que varios componentes tengan exactamente la misma posición, entonces se usa automáticamente un ID de componente interno que garantiza un orden bien definido pero que no se puede modificar.

8 Automatizar asignaciones con productos de Altova

Las asignaciones diseñadas en MapForce se pueden ejecutar en un entorno servidor (incluidos servidores Linux y macOS) y en estos motores de transformación de Altova que ofrecen un rendimiento nivel servidor (y cuyas licencias deben comprarse por separado):

- *RaptorXML Server:* la ejecución de asignaciones en este motor es ideal si el lenguaje de transformación de la asignación es XSLT 1.0, XSLT 2.0, XSLT 3.0 o XQuery (*véase* *Automatización con RaptorXML Server* 428).
- *MapForce Server (o MapForce Server Advanced Edition):* este motor de transformación es adecuado para las asignaciones cuyo lenguaje de transformación sea el motor integrado BUILT-IN*. El lenguaje BUILT-IN es compatible con la mayoría de las características de MapForce, mientras que MapForce Server (y en particular MapForce Server Advanced Edition) ofrece un rendimiento óptimo a la hora de ejecutar asignaciones de datos.

** el lenguaje de transformación BUILT-IN está disponible en las ediciones Professional y Enterprise de MapForce.*

Además, MapForce permite automatizar la generación de código XSLT desde la interfaz de la línea de comandos. Consulte la sección Interfaz de la línea de comandos de MapForce 429 para obtener más información.

8.1 Automatización con RaptorXML Server

RaptorXML Server (en adelante RaptorXML) es el procesador XML y XBRL de alta velocidad y de tercera generación de Altova. Está optimizado para los estándares más recientes y para entornos de computación en paralelo. RaptorXML aprovecha la actual omnipresencia de equipos multinúcleo para ofrecer rapidísimas funciones de procesamiento de datos XML y XBRL.

Altova ofrece dos ediciones distintas de RaptorXML, que se puede descargar e instalar desde la página de descargas de Altova:

- RaptorXML Server es un motor de procesamiento XML ultrarápido compatible con XML, XML Schema, XSLT, XPath, XQuery y muchos otros formatos.
- RaptorXML+XBRL Server ofrece todas las características de RaptorXML Server además de funciones de procesamiento y validación compatibles con toda la gama de estándares XBRL.

Si genera código en XSLT MapForce crea un archivo de procesamiento por lotes llamado `.bat` que se guarda en la carpeta de salida que usted elija. Cuando se ejecuta, este archivo de procesamiento por lotes llama a RaptorXML Server y ejecuta la transformación XSLT en el servidor.
Nota: También puede obtener una vista previa del código XSLT 66 con el motor integrado.

8.2 Interfaz de la línea de comandos de MapForce

La sintaxis general para usar un comando de MapForce en la línea de comandos es:

```
MapForce.exe <filename> [/{target} [[<outputdir>] [/options]]]
```

Para más información sobre cada parámetro del comando, véase la lista a continuación.

Sintaxis de la línea de comandos
Para indicar la sintaxis de la línea de comando se usa la siguiente notación:

Notación	Descripción
Texto sin corchetes ni llaves	Elementos que puede teclear tal y como aparecen
`<Texto entre cuñas>`	Marcador de posición para el que debe indicar un valor
`[Texto entre corchetes]`	Elementos opcionales
`{Texto entre llaves}`	Conjunto de elementos obligatorios; elija uno
Barra vertical `(|)`	Separador para elementos mutuamente excluyentes; elija uno
Puntos suspensivos `(...)`	Elementos que pueden repetirse

- `<filename>`

  El archivo del diseño de la asignación (**`.mfd`**) o del proyecto de la asignación (**`.mfp`**) (*ediciones Professional y Enterprise*) a partir del cual se genera el código.

- `/{target}`

  Indica el lenguaje o entorno de destino para el que se genera el código. Son compatibles los siguientes destinos de generación de código:

  - `/XSLT`
  - `/XSLT2`
  - `/XSLT3`

- `<outputdir>`

  Parámetro opcional que indica el directorio de salida. Si no se indica una ruta de salida, se usará el directorio de trabajo actual. Observe que las rutas relativas lo son al directorio de trabajo actual.

- `/options`

  Las opciones `/options` no se excluyen mutuamente. Se pueden indicar una o más de las siguientes opciones:

  - La opción `/GLOBALRESOURCEFILE <filename>` se puede aplicar si la asignación de datos usa los recursos globales para resolver un archivo de entrada o de salida, rutas a directorios o bases de datos. Para más información, consulte Recursos globales de Altova 432. La

opción `/GLOBALRESOURCEFILE` indica la ruta al archivo XML de los recursos globales. Observe si está indicado `/GLOBALRESOURCEFILE` también ha de estarlo `/GLOBALRESOURCECONFIG`.

- La opción `/GLOBALRESOURCECONFIG <config>` especifica el nombre de la configuración de los recursos globales (*véase la opción anterior*). Si está indicado `/GLOBALRESOURCEFILE` también ha de estarlo `/GLOBALRESOURCECONFIG`.

- La opción `/LOG <logfilename>` genera un archivo de registro en la ruta indicada. La ruta `<logfilename>` puede ser absoluta. Si indica una ruta completa, el directorio debe existir para que se genere el archivo de registro. Si sólo indica el nombre del archivo, este se guardará en el directorio actual del símbolo del sistema de Windows.

Notas

- Las rutas relativas lo son al directorio de trabajo, que es el directorio de la aplicación que llama a MapForce en ese momento. Esto se aplica a la ruta del nombre de archivo **`.mfd`**, directorio de salida, nombre de archivo de registro y nombre de archivo de recurso global.
- No use barras diagonales inversas y comillas de cierre en la línea de comandos (p.ej., **`"C:\Mi directorio\"`**). El analizador sintáctico interpreta estos dos caracteres como comillas dobles literales. Se recomienda evitar el uso de espacios y comillas. Si hay espacios en la línea de comando y necesita las comillas, use la doble barra inversa (p.ej., **`"C:\Mi Directorio\\"`**).

Ejemplos

1) Para iniciar MapForce y abrir la asignación `<filename>.mfd`, use este comando:

```
MapForce.exe <filename>.mfd
```

2) Para generar código XSLT 2.0 y crear también un archivo de registro con el nombre `<logfilename>`, use este comando:

```
MapForce.exe <filename>.mfd /XSLT2 <outputdir> /LOG <logfilename>
```

3) Para generar código XSLT 2.0 teniendo en cuenta la configuración del recurso global `<grconfigname>` del archivo de recurso global `<grfilename>`, use este comando:

```
Mapforce.exe <filename>.mfd /XSLT2 <outputdir> /GLOBALRESOURCEFILE
<grfilename> /GLOBALRESOURCECONFIG <grconfigname>
```

Ejemplos para las ediciones Professional y Enterprise

1) Para generar una aplicación en código C# para Visual Studio 2022 y generar un archivo de registro, use este comando:

```
MapForce.exe <filename>.mfd /CS:VS2022 <outputdir> /LOG <logfilename>
```

2) Para generar una aplicación en C++ usando las opciones de generación de código definidas en **Herramientas | Opciones** y generar un archivo de registro, use este comando:

```
MapForce.exe <filename>.mfd /CPP <outputdir> /LOG <logfilename>
```

3) Para generar una aplicación en C++ para Visual Studio 2022, MSXML, con bibliotecas estáticas, compatible con MFC y sin archivo de registro, use este comando:


```
MapForce.exe <filename>.mfd /CPP:VS2022,MSXML,LIB,MFC
```

4) Para generar una aplicación en C++ para Visual Studio, Xerces, con bibliotecas dinámicas, sin compatibilidad con MFC y con archivo de registro, use este comando:

```
MapForce.exe <filename>.mfd /CPP:VS2022,XERCES,DLL,NoMFC <outputdir> /LOG
<logfilename>
```

5) Para generar una aplicación en código Java y generar un archivo de registro, use este comando:

```
MapForce.exe <filename>.mfd /JAVA <outputdir> /LOG <logfilename>
```

6) Para generar código para todas las asignaciones del proyecto usando el lenguaje y el directorio de salida definidos en las opciones de la carpeta (de cada carpeta dentro del proyecto), use este comando:

```
MapForce.exe <filename>.mfp /GENERATE /LOG <logfilename>
```

7) Para generar código Java para todas las asignaciones del archivo del proyecto, use este comando:

```
MapForce.exe <filename>.mfp /JAVA /LOG <logfilename>
```

Con esta opción se ignora el lenguaje de generación de código definido en las opciones de la carpeta y se usa Java para todas las asignaciones.

8) Para indicar los archivos de entrada y de salida en la línea de comandos para una asignación en Java compilada con anterioridad, use este comando:

```
java -jar <mappingfile>.jar /InputFileName <inputfilename> /OutputFileName
<outputfilename>
```

Los parámetros `/InputFileName` y `/OutputFileName` son nombres de componentes especiales de entrada de la asignación de datos de MapForce que permiten usar parámetros en la ejecución de la línea de comandos (véase Pasar parámetros a la asignación 147).

9) Para compilar una asignación en un archivo de ejecución de MapForce Server para la versión de MapForce 2024 y eliminar las firmas XML, use este comando:

```
MapForce.exe <filename>.mfd /COMPILE:NOXMLSIGNATURES
<outputdir> /MFXVERSION:2024 /LOG <logfilename>
```

9 Recursos globales de Altova

Los recursos globales de Altova son alias para recursos de archivo, carpeta y base de datos. Cada alias puede tener varias configuraciones y cada configuración se corresponde con un solo recurso. Esto quiere decir que puede alternar entre configuraciones al usar recursos globales. Por ejemplo, podría crear un recurso de base de datos con dos configuraciones: `desarrollo` y `producción`. Puede cambiar de una configuración a otra en función de sus necesidades.

Los recursos globales se pueden usar en distintas aplicaciones de Altova (*véase la lista más abajo*).

Recursos globales en otros productos de Altova

Una vez los haya guardado como recursos globales, puede usar esos archivos, carpetas o conexiones de base de datos en varias de las aplicaciones de Altova. Por ejemplo, si suele tener que abrir el mismo archivo (o carpeta o conexión de base de datos) en distintas aplicaciones de escritorio de Altova puede que le resulte más cómodo definirlo como un recurso global. Así, si quiere cambiar la ruta de acceso solo tiene que hacerlo en un sitio. Por el momento, puede definir y usar recursos globales en estos productos de Altova:

- Altova Authentic
- DatabaseSpy
- MobileTogether Designer
- MapForce
- StyleVision
- XMLSpy
- FlowForce Server
- MapForce Server
- RaptorXML Server/RaptorXML+XBRL Server

Apartados de esta sección

En esta sección explicamos cómo crear y configurar distintos tipos de recursos globales. Estos son los apartados de esta sección:

9.1 Configurar recursos globales - parte 1

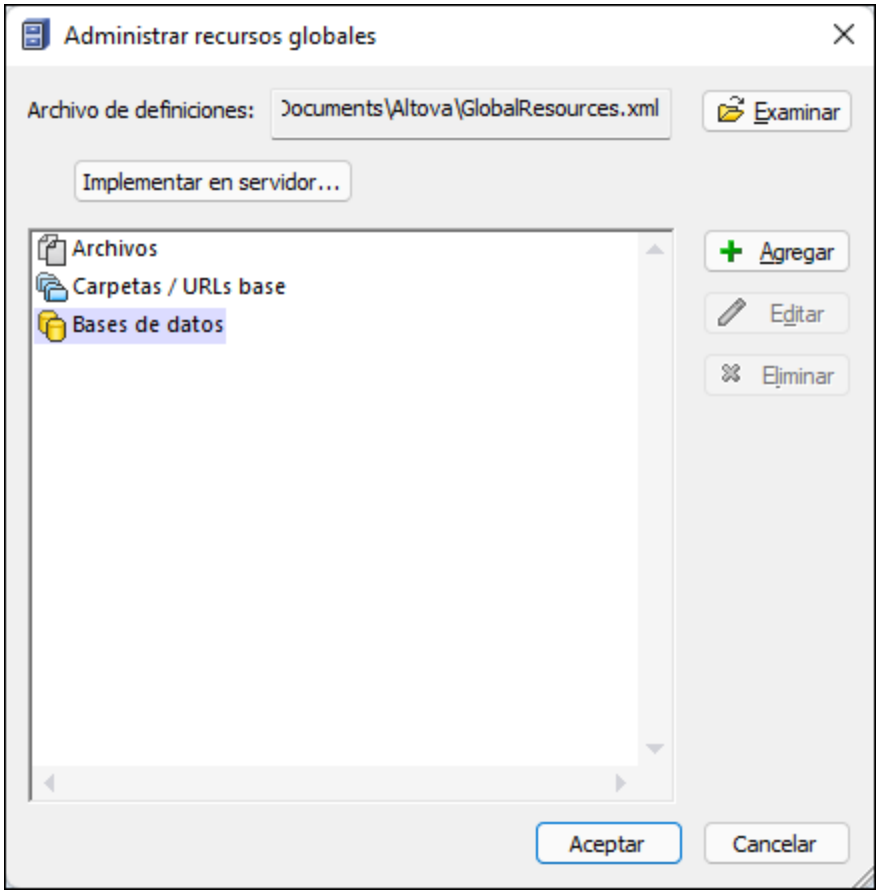
La configuración de los recursos globales tiene dos pasos: (I) crear el recurso global en el cuadro de diálogo **Administrar recursos globales** (*imagen siguiente*) y (ii) definir las propiedades de este recurso global en el cuadro de diálogo **Recursos globales**. La segunda parte se explica en el apartado siguiente 435.

Los recursos globales de Altova se definen en el cuadro de diálogo **Administrar recursos globales**, al que se puede acceder de dos formas:

- Haga clic en el comando de menú **Herramientas | Recursos globales**.
- Haga clic en el icono **Administrar recursos globales** de la barra de herramientas Recursos globales (*imagen siguiente*).

El archivo de definiciones de recursos globales

La información sobre los recursos globales se almacena en un archivo XML llamado archivo de definiciones de recursos globales. Este archivo se crea y guarda cuando se define el primer recurso global en el cuadro de diálogo **Administrar recursos globales** (*imagen siguiente*).



Cuando se abre el cuadro de diálogo **Administrar recursos globales** por primera vez, el nombre y la ubicación predeterminados del archivo de definiciones de recursos globales aparece en el cuadro de texto *Archivo de definiciones* (*imagen anterior*):

C:\Usuarios\<usuario>\Documentos\Altova\GlobalResources.xml.

Este archivo se define como archivo predeterminado de definiciones de recursos globales para todas las aplicaciones de Altova. Así puede guardar en este archivo recursos globales desde cualquier aplicación de Altova y el recurso global estará a disposición de todas las aplicaciones de Altova. Para definir y guardar un recurso global en el archivo de definiciones, añada el recurso global en el cuadro de diálogo **Administrar recursos globales** y haga clic en **Aceptar**.

Para seleccionar un archivo de definiciones de recursos globales ya existente como archivo de definiciones activo, búsquelo con el botón **Examinar** del cuadro de texto *Definiciones* (*imagen anterior*).

En el cuadro de diálogo **Administrar recursos globales** también permite editar y eliminar recursos globales.

Notas:

- Puede asignar cualquier nombre al archivo de definiciones de recursos globales y guardarlo en cualquier ubicación a la que tengan acceso sus aplicaciones de Altova. Después en las aplicaciones de Altova sólo tiene que definir este archivo como archivo de definiciones de recursos globales (en el cuadro de texto *Definiciones*). Ahora el recurso global se puede usar en distintos productos de Altova con una sola definición para todos ellos.
- Si lo prefiere, también puede crear varios archivos de definiciones de recursos globales. Sin embargo, en cada aplicación de Altova sólo puede estar activo un archivo de definiciones y, por tanto, sólo las definiciones de ese archivo estarán a disposición de la aplicación. También puede restringir la disponibilidad de los recursos o ampliarla a varios productos.

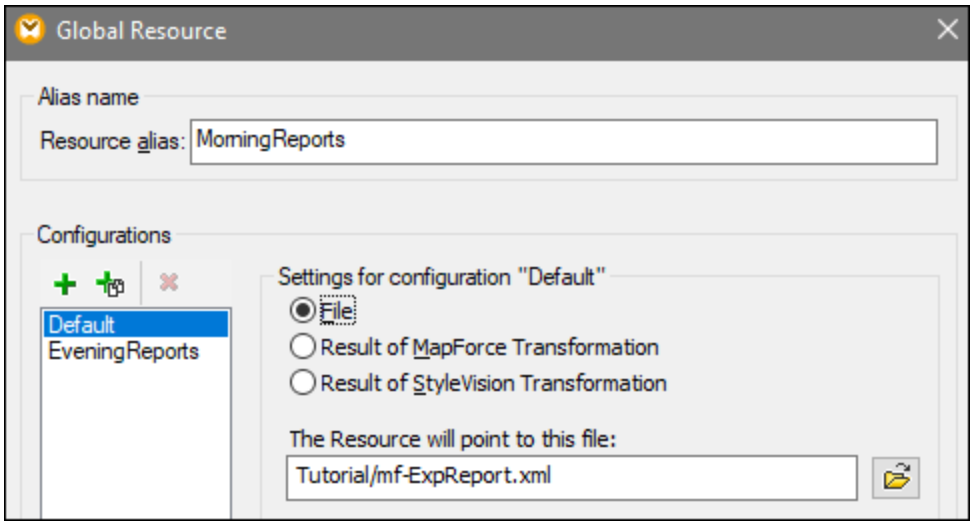
9.2 Configurar recursos globales - parte 2

The second part of the global resource setup consists in defining properties of a global resource in the **Global Resource** dialog box. The properties depend on the type of a global resource (*see subsections below*). You can access the **Global Resource** dialog box by clicking the **Add** button in the Manage Global Resources dialog box 433.

To find out more about setting up different types of global resources, see the following examples: XML Files as Global Resources 438, Folders as Global Resources 440.

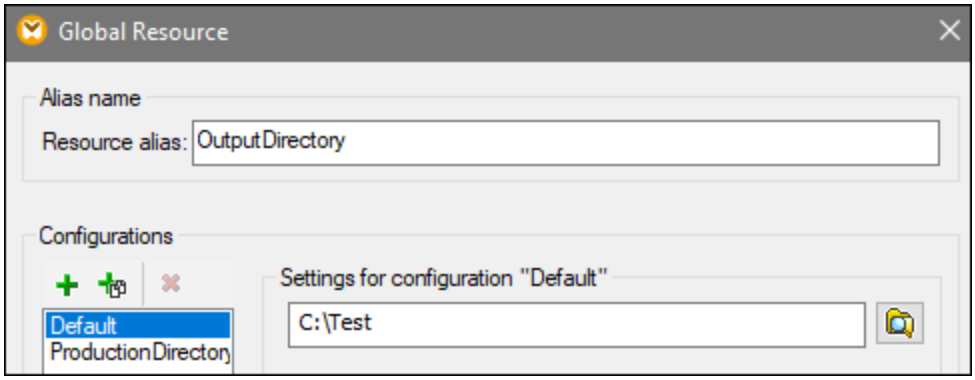
Files

The file-specific properties are shown in the **Global Resource** dialog box below. The setup has three major parts: (i) the name of the file, (ii) the location of this file, and (iii) the list of configurations defined for this file alias.



Folders

The folder-specific properties are shown in the **Global Resource** dialog box below. The setup has three major parts: (i) the name of the folder, (ii) the location of this folder, and (iii) the list of configurations defined for this folder alias.



Global Resource dialog icons

	Add Configuration: Pops up the Add Configuration dialog in which you enter the name of the configuration to be added.
	Add Configuration as Copy: Pops up the Add Configuration dialog in which you can enter the name of the configuration to be created as a copy of the selected configuration.
	Delete: Deletes the selected configuration.
	Open: Browse for the file to be created as the global resource.
	Open: Browse for the folder to be created as the global resource.

Global resource setup: General procedures

The broad procedure of creating and configuring global resources is described below:

1. Click the toolbar button (**Manage Global Resource**). Alternatively, go to the **Tools** menu and click **Global Resources**.
2. Click **Add** and select the resource type you wish to create (file, folder, database). The **Global Resource** dialog box will appear.
3. Enter a descriptive name in the **Resource alias** text box (e.g., `InputFile`).
4. Setting up the default configuration depends on the type of the global resource: (i) For a file or folder, browse for the file or folder to which this resource should point by default; (ii) for a database connection, click **Choose Database** and follow the Database Connection Wizard to connect to the database . This database connection will be used by default when you run the mapping.

5. If you need an additional configuration (e.g., an additional output folder), click the button in the **Global Resource** dialog box, enter the name of this configuration, and specify the path to this configuration.
6. Repeat the previous step for each additional configuration required.

Note: Database connections are supported as global resources only in MapForce Professional and Enterprise editions.

9.3 Archivos XML como recursos globales

En este apartado explicamos cómo usar archivos XML como recursos globales. Puede haber situaciones en las que quiera cambiar un archivo XML de entrada varias veces al día. Por ejemplo, imaginemos que todas las mañanas ejecuta una asignación en concreto y genera un informe usando como entrada un archivo XML, y todas las tardes ejecuta la misma asignación para generar otro reporte, pero este usa un archivo XML distinto como entrada. Aquí es donde pueden ser útiles los recursos globales: en lugar de editar la asignación varias veces al día (o tener varias copias de la misma), podría configurar la asignación para que que lea un archivo definido como recurso global (lo que se conoce como *alias de archivo*). En este ejemplo, nuestro alias de archivo tiene dos configuraciones:

1. `Default`, que daría un archivo XML "de mañana" como entrada para la asignación.
2. `EveningReports`, que daría un archivo XML "de tarde" como entrada para la asignación.

Para crear y configurar el alias de archivo siga estos pasos que explicamos a continuación.

Paso 1: crear el recurso global

Primero debe crear un alias de archivo. Para ello siga estos pasos:

1. Haga clic en el botón  de la barra de herramientas (**Administrar recursos globales**). Si lo prefiere, vaya al menú **Herramientas** y haga clic en **Recursos globales**.
2. Haga clic en **Agregar | Archivo** e introduzca un nombre en el cuadro de texto **Alias del recurso**. En este ejemplo podríamos usar `MorningReports`.
3. Haga clic en **Examinar** y seleccione **`Tutorial\mf-ExpReport.xml`**.
4. Haga clic en la sección **Agregar configuración**  y llame a la segunda configuración `EveningReports`.
5. Haga clic en **Examinar** y seleccione **`Tutorial\mf-ExpReport2.xml`**.

Paso 2: usar el recurso global en la asignación

Ahora puede usar el recurso global que acaba de crear en la asignación. Para que esta lea el recurso global, siga estos pasos:

1. Abra la asignación **`Tutorial\Tut-ExpReport.mfd`**.
2. Haga doble clic el encabezado del componente de origen para abrir el cuadro de diálogo **Configuración del componente**.
3. Junto a **Archivo XML de entrada**, haga clic en **Examinar** y seleccione `MorningReports`. Haga clic en **Abrir**.
4. Se abrirá el cuadro de diálogo **Configuración del componente**. La ruta del archivo XML de entrada ahora se llama **`altova://file_resource/MorningReports`**, lo que indica que la ruta está usando un recurso global.

Paso 3: ejecutar la asignación con la configuración deseada

Ahora puede alternar entre los dos archivos XML de entrada antes de ejecutar la asignación:

- Para usar **`mf-ExpReport.xml`** como directorio de entrada, seleccione el elemento de menú **Herramientas | Configuración activa | Default**.
- Para usar **`mf-ExpReport2.xml`** como directorio de salida seleccione el elemento de menú **Herramientas| Configuración activa | ProductionDirectory**.

También puede seleccionar la configuración que desee en la lista desplegable de los **recursos globales** (*imagen siguiente*).

Para generar una vista previa del resultado de la asignación con cualquiera de las configuraciones haga clic en el panel **Resultados**.

9.4 Carpetas como recursos globales

En este apartado explicamos cómo usar las carpetas como recursos globales. Imaginemos que a veces necesita generar la salida de la asignación en varios directorios distintos. Con los recursos globales esto puede hacerse creando un alias de carpeta con dos configuraciones:

1. Configuración `Default`: genera la salida en **`C:\Test`**.
2. Configuración `Production`: genera la salida en **`C:\Production`**.

En los siguientes pasos explicamos cómo generar los resultados en distintas carpetas.

Paso 1: crear el recurso global

Primero debe crear un alias de carpeta. Para ello siga estos pasos:

1. Haga clic en el botón  de la barra de herramientas (**Administrar recursos globales**). Si lo prefiere, vaya al menú **Herramientas** y haga clic en **Recursos globales**.
2. Haga clic en **Agregar | Carpeta** e introduzca un nombre en el cuadro de texto **Alias del recurso**. En este ejemplo podríamos usar `OutputDirectory`.
3. Haga clic en **Examinar** y seleccione la siguiente carpeta: C:\Test. Asegúrese primero de que esta carpeta ya existe en su sistema operativo.
4. Haga clic en  e introduzca un nombre para la nueva configuración En este ejemplo, `ProductionDirectory`.
5. Haga clic en **Examinar** y seleccione la carpeta **`C:\Production`**. Asegúrese primero de que esta carpeta ya existe en su sistema operativo.

Paso 2: usar el recurso global en la asignación

Ahora ha creado el recurso global. Sin embargo, la asignación todavía no lo está usando. Para modificar la asignación para que use el alias de carpeta definido con anterioridad (Recurso global), siga estos pasos:

1. Abra la asignación **`Tutorial\Tut-ExpReport.mfd`**.
2. Haga doble clic el encabezado del componente de destino para abrir el cuadro de diálogo **Configuración del componente**.
3. Haga clic en **Cambiar a recursos globales** y luego en **Guardar**.
4. Guarde el archivo XML de salida como **`Output.xml`**. La ruta del archivo XML de salida ahora es **`altova://folder_resource/OutputDirectory/Output.xml`**, lo que indica que la ruta está usando un recurso global.

Paso 3: ejecutar la asignación con la configuración deseada

Ahora puede cambiar fácilmente el la carpeta de salida de la asignación antes de ejecutarla:

- Para usar **`C:\Test`** como directorio de salida, seleccione el elemento de menú **Herramientas| Configuración activa | Default**.
- Pra usar **`C:\Production`** como directorio de salida seleccione el elemento de menú **Herramientas| Configuración activa | ProductionDirectory**.

La salida de la asignación se escribe por defecto como un archivo temporal, a no ser que configure MapForce explícitamente para que escriba la salida en archivos permanentes. Para configurar MapForce para que genere archivos permanentes en lugar de temporales, siga estos pasos:

1. En el menú **Herramientas** haga clic en el comando **Opciones**.
2. En la sección **General**, seleccione la opción **Escribir directamente en archivos de salida finales**.

10 Catálogos en MapForce

MapForce es compatible con un subconjunto del mecanismo de catalogación XML OASIS. El mecanismo de catalogación permite a MapForce recuperar de carpetas locales del usuario los esquemas (y hojas de estilos y otros archivos) usados con frecuencia. Esto incrementa la velocidad global de procesamiento, permite al usuario trabajar sin conexión (es decir, sin estar conectado a una red) y mejora la portabilidad de los documentos (porque los identificadores URI se tienen que cambiar sólo en los archivos de catálogo).

A continuación describimos cómo funciona el mecanismo de catalogación en MapForce.

Para más información consulte la especificación XML Catalogs.

10.1 Funcionamiento de los catálogos

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

DTDs

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

Para los esquemas más utilizados el identificador **PUBLIC** suele estar predefinido y, por tanto, sólo hace falta que el URI del archivo de catálogo asigne el identificador **PUBLIC** a la copia local correcta. Cuando se analiza el documento XML, se lee el identificador **PUBLIC** del documento. Si se encuentra este identificador en un archivo de catálogo, se buscará la URL correspondiente del archivo de catálogo y se leerá el esquema desde esta ubicación. Por ejemplo, imaginemos que abrimos este archivo SVG en MapForce:

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

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

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

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

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

Esquemas XML

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

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

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

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

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

10.2 Estructura de los catálogos de MapForce

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

Extracto de RootCatalog.xml

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

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

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

Variables de ubicación

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

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

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

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

10.3 Personalizar catálogos

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

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

Tenga en cuenta que:

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

A partir de su versión de 2014 MapForce cumple escrupulosamente con la especificación de catálogos XML XML Catalogs specification (OASIS Standard V1.1, 7 October 2005) Esta especificación separa estrictamente las consultas por identificador externo (las realizadas con un ID público o de sistema) de las búsquedas por URI (los URI que no son ID públicos ni de sistema). Por tanto, los URIs de espacios de nombres deben considerarse simplemente como URIs (no como IDs públicos o del sistema) y deben usarse como búsquedas por URI y no como consultas por identificador externo. En las versiones de MapForce previas a la de 2014 los URIs de espacios de nombres de esquemas se traducían con asignaciones **<public>**. Sin embargo, a partir de la versión 2014 es necesario utilizar asignaciones **<uri>**.

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

Cómo encuentra MapForce un esquema de referencia

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

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

A continuación damos los pasos que sigue MapForce de forma secuencial para encontrar un esquema de referencia. El esquema se carga en el primer paso que se ejecuta correctamente.

1. Consulte en el catálogo la parte del URI del valor **xsi:schemaLocation**. Si se encuentra una asignación, y esto incluye las asignaciones **rewriteURI**, use el URI resultante para cargar el esquema.
2. Consulte en el catálogo la parte del espacio de nombres del valor **xsi:schemaLocation**. Si se encuentra una asignación, y esto incluye las asignaciones **rewriteURI**, use el URI resultante para cargar el esquema.
3. Use la parte URI del valor **xsi:schemaLocation** para cargar el esquema.

Especificaciones de XML Schema

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

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

10.4 Variables de entorno

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

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

r%	
%WindowsFolder%	Ruta de acceso completa de la carpeta Windows del usuario actual.
%SystemFolder%	Ruta de acceso completa de la carpeta System del usuario actual.
%LocalAppDataFolder%	Ruta de acceso completa al directorio del sistema de archivos que sirve como repositorio de datos para aplicaciones locales (no roaming).
%MyPicturesFolder%	Ruta de acceso completa a la carpeta Mis imágenes.

11 Comandos de menú

En esta sección se describirán los comandos de menú en MapForce. Estos son los comandos de menú disponibles:

11.1 Archivo

En este apartado puede consultar todos los comandos del menú **Archivo**.

- Nuevo

  Crea un documento de asignación nuevo. En las ediciones Professional y Enterprise también puede crear un proyecto de asignación de datos (`.mfp`).

- Abrir

  Abre el archivo de diseño de asignación guardado anteriormente (`.mfd`). En las ediciones Professional y Enterprise también abre proyectos de asignación de datos (`.mfp`).

- Guardar, Guardar como, Guardar todos

  La opción **Guardar** guarda la asignación activa usando el nombre de archivo activo en ese momento. La opción **Guardar como** guarda la asignación activa con otro nombre o permite al usuario indicar un nombre nuevo si es la primera vez que se guarda el archivo. La opción **Guardar todos** guarda todos los archivos de asignación que están abiertos.

- Volver a cargar

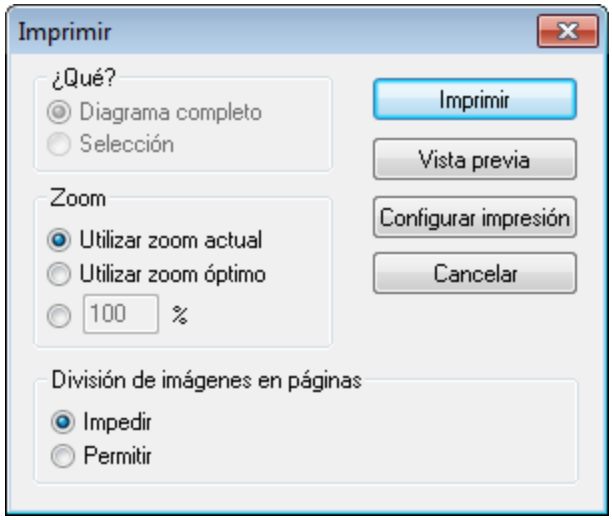
  Vuelve a cargar el archivo de asignación activo. Al volver a cargarlo, se revierten los últimos cambios.

- Cerrar, Cerrar todos

  La opción **Cerrar** cierra el archivo de asignación activo. La opción **Cerrar todos** cierra todos los archivos de asignación activos. Aparece un aviso preguntando si desea guardar los archivos que están sin guardar.

- Imprimir, Vista previa de impresión, Configurar impresión

  La opción **Imprimir** abre el cuadro de diálogo **Imprimir**, desde donde puede imprimir la asignación (*imagen siguiente*). La opción **Utilizar zoom actual** conserva el factor de zoom actual La opción **Utilizar zoom óptimo** ajusta el tamaño de la asignación al tamaño de la página. También puede especificar el factor de zoom en porcentaje. Las barras de desplazamiento de los componentes de la asignación no se imprimen. También puede permitir o impedir la división de las imágenes en varias páginas.



El comando **Vista previa de impresión** abre el cuadro de diálogo **Imprimir** con las mismas opciones que se describen más arriba. **Configurar impresión** abre el cuadro de diálogo **Configurar impresión**, donde puede seleccionar la impresora que desea usar y definir la configuración del papel.

- Validar asignación

  El comando **Validar asignación** comprueba si todas las asignaciones (conectores) son válidas y emite advertencias y errores de validación. Para más detalles consulte Validación 64.

- Configurar asignación

  Abre el cuadro de diálogo Cambiar configuración de la asignación 76, donde puede definir las opciones de configuración del documento activo.

- Abrir el gestor de credenciales (*Enterprise Edition*)

  Abre el **Gestor de credenciales**, donde puede gestionar las credenciales que necesite para las asignaciones que funcionan con autenticación HTTP básica o autorización OAuth 2.0.

- Generar código en el lenguaje seleccionado, Generar código en

  El comando **Generar código en el lenguaje seleccionado** genera código en el lenguaje seleccionado en la barra de herramientas. El comando **Generar código en <idioma>** permite generar código en XSLT 1-3 (*todas las ediciones*), XQuery, Java, C# y C++ (*ediciones Professional y Enterprise*). Al hacer clic en estos comandos se abre el cuadro de diálogo **Buscar carpeta**, donde puede seleccionar la ubicación de los archivos generados. Los nombres de los archivos generados se define en el cuadro de diálogo Configurar asignación 76.

  Para más información sobre los lenguajes de transformación disponibles consulte Lenguajes de transformación 17. Para más información sobre cómo generar código, consulte Generación de código 66.

- Compilar en archivo de ejecución de MapForce Server (*ediciones Professional y Enterprise*)

Genera un archivo que se puede ejecutar directamente en MapForce Server para ejecutar la transformación de la asignación de datos.

- Implementar en FlowForce Server (*ediciones Professional y Enterprise*)

  Implementa la asignación activa en FlowForce Server.

- Generar documentación (*ediciones Professional y Enterprise*)

  Genera documentación para los proyectos de asignación en varios formatos de salida.

- Archivos recientes

  Muestra una lista con archivos abiertos recientemente.

- Salir

  Cierra la aplicación. Antes aparece un aviso preguntando si desea guardar los archivos que estén sin guardar.

11.2 Edición

En este apartado puede consultar todos los comandos del menú **Edición**. La mayoría de los comandos del menú **Edición** se habilitan cuando se abre el panel **Resultados** o cuando se consulta la vista previa de código XSLT en el panel **XSLT**.

- Deshacer

  MapForce ofrece un número ilimitado de operaciones deshacer, que puede usar para retroceder en todos los cambios realizados. También puede usar el icono  .

- Rehacer

  Este comando permite rehacer cambios anulados anteriormente. Puede avanzar y retroceder por el historial de cambios usando estos dos comandos. También puede usar el icono  .

- Buscar

  Permite buscar una cadena de texto en los paneles **XSLT**, **XSLT2**, **XSLT3**, **XQuery** (ediciones *Professional y Enterprise*) y **Resultados**. También puede usar el icono  .

- Buscar siguiente

  Busca la siguiente instancia del término de búsqueda. También puede usar el icono  .

- Buscar anterior

  Busca la instancia anterior del término de búsqueda. También puede usar el icono  .

- Cortar / Copiar / Pegar / Eliminar

  Estos comandos de edición estándar de Windows permiten cortar, copiar, etc. componentes y funciones de la ventana de asignación.

- Seleccionar todo

  Selecciona todos los componentes del panel **Asignación** o todo el texto de los paneles **XSLT**, **XSLT2**, **XSLT3**, **XQuery** (ediciones *Professional y Enterprise*) y **Resultados**.

11.3 Insertar

En este apartado puede consultar todos los comandos del menú **Insertar**.

- Archivo o esquema XML

  Inserta un archivo de esquema XML o de instancia XML a la asignación. Si selecciona un archivo XML sin referencia a un esquema, deberá generar automáticamente el esquema XML correspondiente113. Si selecciona un archivo de esquema XML, aparece un aviso preguntando si desea incluir un archivo XML de instancia que aporte los datos para la vista previa. También puede agregar un archivo XML/XSD con el comando de la barra de herramientas  .

- Base de datos (*ediciones Professional y Enterprise*)

  Inserta una base de datos como componente de origen o destino. También puede agregar un componente de BD con el comando de la barra de herramientas  . En MapForce Enterprise Edition también puede agregar BDs NoSQL como componentes.

- EDI (*Enterprise Edition*)

  Inserta un documento EDI como componente de origen o destino. También puede agregar un componente EDI con el comando de la barra de herramientas  .

- Archivo de texto (*ediciones Professional y Enterprise*)

  Inserta un archivo plano (es decir, un archivo de texto CSV o de longitud fija) en la asignación de datos. Ambos tipos de archivo pueden usarse como componente de origen o de destino. También puede agregar un archivo de texto con el comando de la barra de herramientas  . MapForce Enterprise Edition también permite procesar archivos de texto con FlexText.

- Función de servicio web (*Enterprise Edition*)

  Inserta en la asignación una llamada a un servicio web. También puede agregar un servicio web con el comando de la barra de herramientas  .

- Archivo Excel 2007+ (*Enterprise Edition*)

  Inserta un archivo Microsoft Excel 2007+ (Office Open XML) (**`.xlsx`**) como componente de origen o destino. Si no tiene Excel 2007+ instalado, esto no le impide crear asignaciones de datos entre archivos Excel 2007+. El único impedimento es que no podrá consultar la vista previa de resultados en el panel **Resultados**. Sí podrá guardar los resultados con el comando **Guardar el archivo de salida** del menú **Resultados**. También puede agregar un archivo de Excel con el comando de la barra de herramientas  .

- Documento XBRL (*Enterprise Edition*)

  Inserta un documento de instancia XBRL o una taxonomía XBRL en la asignación. También puede agregar un componente XBRL con el comando de la barra de herramientas  .

- Archivo o esquema JSON (*Enterprise Edition*)

  Inserta un archivo o un esquema JSON en la asignación. También puede agregar un componente JSON con el comando de la barra de herramientas  .

- Archivo de Protocol Buffers (*Enterprise Edition*)

  Añade a la asignación un archivo binario cifrado en formato Protocol Buffers. También puede agregar un componente Protocol Buffers con el comando de la barra de herramientas  .

- Insertar documento PDF (*Enterprise Edition*)

  Agrega un documento PDF. También puede insertar un documento PDF a través de la barra de herramientas.

- Insertar componente de entrada

  Si en el área de asignación está visible una asignación de datos, este comando inserta un componente de entrada. Si en el área de asignación está visible una función definida por el usuario, este comando agrega un componente de entrada en la función. Para más información consulte Pasar parámetros a la asignación 147 y Parámetros en funciones definidas por el usuario 211. También puede agregar un componente de entrada simple con el comando de la barra de herramientas  .

- Insertar componente de salida

  Si en el área de asignación está visible una asignación de datos, este comando inserta un componente de salida. Si en el área de asignación está visible una función definida por el usuario, este comando agrega un componente de salida en la función. Para más información véase Obtener valores de cadena de una asignación 156 y Parámetros en funciones definidas por el usuario 211. También puede agregar un componente de salida simple con el comando de la barra de herramientas  .

- Constante

  Inserta una constante que aporta datos fijos a un conector de entrada. Los datos se introducen en un cuadro de diálogo que aparece cuando se empieza a crear la constante. Puede elegir entre estos tipos de datos: `cadena`, `número`, `demás opciones`. También puede agregar una constante con el comando de la barra de herramientas  .

- Variable

  Inserta una variable 160 intermedia que equivale a una función definida por el usuario no inline. Las variables son componentes estructurales, sin archivos de instancia, que sirven para simplificar el proceso de asignación. También puede agregar una variable con el comando de la barra de herramientas  .

- Combinar (*ediciones Professional y Enterprise*)

  El componente de combinación permite combinar datos en los modos SQL y non-SQL. También puede agregar un componente de combinación con el comando de la barra de herramientas  .

- Componente de ordenación: nodos/filas

Inserta un componente que permite ordenar nodos (véase Ordenar datos[172]). También puede agregar una variable con el comando de la barra de herramientas  .

- Filtro: nodos/filas

  Inserta un componente de filtro que filtra los datos de cualquier componente compatible con MapForce, incluidas BDs. Consulte la sección Filtros y condiciones[178] para obtener más información. También puede agregar un filtro con el comando de la barra de herramientas  .

- WHERE/ORDER de SQL/NoSQL (*ediciones Professional y Enterprise*)

  Inserta un componente que filtra datos de la base de datos de forma condicional. También puede acceder al componente WHERE/ORDER de SQL/NoSQL con el comando de la barra de herramientas  .

- Asignación de valores

  Inserta un componente que transforma un valor de entrada en un valor de salida usando una tabla de búsqueda. Es un componente muy práctico si necesita asignar un conjunto de valores a otro conjunto de valores (p.ej. números de meses en nombres de meses). Para más información consulte Usar asignaciones de valores[184]. También puede agregar este componente con el comando de la barra de herramientas  .

- Condición If-Else

  Inserta un componente de tipo `If-Else` para filtrar valores simples de forma condicional. Consulte la sección Filtros y condiciones[178] para obtener más información. También puede agregar un componente de tipo `If-Else` con el comando de la barra de herramientas  .

- Excepción (*ediciones Professional y Enterprise*)

  Este componente permite interrumpir el proceso de asignación cuando se cumple determinada condición. También puede agregar un componente de excepción con el comando de la barra de herramientas  . En Enterprise Edition este componente también permite definir mensajes de error (fault) en proyectos de asignación WSDL.

11.4 Componente

En este apartado puede consultar todos los comandos del menú **Componente**.

- Cambiar de elemento raíz

  Permite cambiar el elemento raíz del documento de instancia XML.

- Editar la definición del esquema en XMLSpy

  Para para poder editar un esquema en Altova XMLSpy, debe hacer clic en un esquema XML y seleccionar la opción **Editar la definición del esquema en XMLSpy**.

- Editar la configuración de FlexText (*Enterprise Edition*)

  Abre el módulo FlexText donde puede editar un archivo de FlexText ya existente.

- Agregar, quitar o editar objetos de la base de datos (*ediciones Professional y Enterprise*)

  Permite agregar, eliminar o cambiar los objetos de la base de datos dentro del componente de base de datos.

- Crear asignación a EDI X12 997 (*Enterprise Edition*)

  La confirmación funcional X12 997 informa del estado del intercambio EDI. Todos los errores encontrados durante el procesamiento del documento aparecen en esta confirmación. MapForce puede generar documentos X12 997 automáticamente en el panel Asignación.

- Crear asignación a EDI X12 999 (*Enterprise Edition*)

  El conjunto de transacciones de la confirmación de implementación X12 999 notifica el incumplimiento de las normas de implementación HIPAA y errores de aplicación. MapForce puede generar automáticamente un componente X12 999 y crear automáticamente las conexiones de asignación necesarias.

- Actualizar (*ediciones Professional y Enterprise*)

  Vuelve a cargar la estructura del componente de BD activo desde la base de datos.

- Agregar delante/detrás un duplicado de entrada

  Inserta una copia del elemento seleccionado antes o después del elemento seleccionado. Los elementos de entrada duplicados no se pueden utilizar como origen de datos. Para más información consulte Duplicar elementos de entrada[40].

- Quitar el duplicado

  Quita el elemento duplicado seleccionado.

- Comentario/Instrucción de procesamiento

  Esta opción permite insertar comentarios e instrucciones de procesamiento[123] en componentes XML.

- Escribir contenido como sección CDATA

Este comando crea una sección CDATA[123] trata como unidades completas partes del documento que normalmente se interpretarían como marcado.

- Acciones de tablas de BD (*ediciones Professional y Enterprise*)

  Permite definir qué acciones de tabla se deben llevar a cabo con los datos asignados en cada tabla de la BD de destino.

- Consultar la base de datos (*ediciones Professional y Enterprise*)

  Crea una instrucción SELECT basada en la tabla o en el campo activos en el componente de base de datos. Este comando se habilita al hacer clic una vez en la tabla o en el campo. Así la instrucción SELECT se introduce automáticamente en la ventana **Select**.

- Alinear la estructura a la izquierda

  Coloca todos los elementos de un componente en el lado izquierdo de la ventana.

- Alinear la estructura a la derecha

  Coloca todos los elementos de un componente en el lado derecho de la ventana.

- Editar comentario

  Si en su asignación hay un componente de comentario, puede editarlo haciendo clic en él y seleccionando el comando **Editar comentario**. También puede hacer doble clic dentro del componente de comentario y editar el texto directamente en la caja del comentario. Para más información sobre los componentes de comentarios y sus tipos, consulte Comentario[34].

- Propiedades

  Abre un cuadro de diálogo que muestra las opciones de configuración del componente seleccionado. Consulte Cambiar la configuración de los componentes[39].

11.5 Conexión

En este apartado puede consultar todos los comandos del menú **Conexión**.

- Conectar automáticamente los secundarios equivalentes

  Activa o desactiva la opción **Conexión automática de secundarios**. Para más información sobre conexiones y sus tipos consulte Conexiones 46.

- Configurar la conexión de secundarios equivalentes

  Le ayuda a definir las conexiones de secundarios equivalentes. Para más detalles, consulte Conexiones de secundarios equivalentes 53.

- Conectar los secundarios equivalentes

  Este comando permite crear varios conectores para los elementos que tienen el mismo nombre en el esquema de origen y en el de destino. Las opciones definidas en este cuadro de diálogo se aplican cuando esté activo el icono  (**Conexión automática de secundarios**) de la barra de herramientas. Para más detalles, consulte Conexiones de secundarios equivalentes 53.

- Basada en destino (estándar)

  Cambia el tipo de conector a una asignación estándar. Para más información consulte Conexiones basadas en el destino vs. conexiones basadas en el origen 50.

- Copia total (copia de los elementos secundarios)

  Crea conexiones para todos los elementos secundarios equivalentes. La ventaja principal de las conexiones de copia total es que simplifican visualmente el área de trabajo: En vez de múltiples conexiones se crea una sola conexión, representada por una línea gruesa. Consulte también Conexiones de copia total 55.

- Basada en origen (contenido mixto)

  Cambia el tipo de conector a una asignación basada en origen o de contenido mixto (texto y nodos secundarios) en el mismo orden que el del archivo XML de *origen*. Para más información consulte Conexiones basadas en el origen 50.

- Propiedades

  Abre el cuadro de diálogo **Configuración de la conexión** donde puede definir los tipos de conexión y la configuración de la anotación. Para más información consulte Configuración de la conexión 56.

11.6 Función

En este apartado puede consultar todos los comandos del menú **Función**.

- Crear una función definida por el usuario

  Crea una función nueva definida por el usuario 205 (UDF por sus siglas en inglés). También puede crear una función definida por el usuario con el comando de la barra de herramientas .

- Crear una función definida por el usuario a partir de la selección

  Crea una función nueva definida por el usuario basada en los elementos que están seleccionados en el área de asignación. Para más detalles consulte Crear funciones definidas por el usuario 208. También puede crear una función definida por el usuario a partir de la selección con el comando de la barra de herramientas .

- Configuración de la función

  Abre el cuadro de diálogo de configuración de la función definida por el usuario que está activa para poder cambiar su configuración. Para más detalles consulte Editar funciones definidas por el usuario 210.

- Quitar función

  Elimina la función definida por el usuario que está activa (siempre y cuando el contexto lo permita).

- Insertar componente de entrada

  Si en el área de asignación está visible una asignación de datos, este comando inserta un componente de entrada (*véase Entradas simples* 148). Si en el área de asignación está visible una función definida por el usuario, este comando agrega un componente de entrada en la función (*véase Parámetros en funciones definidas por el usuario* 211). También puede insertar un componente de entrada simple con el comando de la barra de herramientas .

- Insertar componente de salida

  Si en el área de asignación está visible una asignación de datos, este comando inserta un componente de salida (*véase Salidas simples* 157). Si en el área de asignación está visible una función definida por el usuario, este comando agrega un componente de salida a la función función nueva definida por el usuario 205. También puede insertar un componente de salida simple con el comando de la barra de herramientas .

11.7 Resultados

En este apartado puede consultar todos los comandos del menú **Resultados**.

- XSLT 1.0, XSLT 2.0, XSLT 3.0, XQuery, Java, C#, C++, Motor de ejecución integrado

  Establece el lenguaje de transformación en el que se debe ejecutar la asignación de datos. Los lenguajes de transformación disponibles dependen de la edición de MapForce. Para más detalles consulte Seleccionar un lenguaje de transformación[17]. También puede seleccionar un lenguaje de transformación a través de la barra de herramientas.

- Validar archivo de salida

  Valida el archivo XML de salida con el esquema al que se hace referencia. Consulte Validación[64].

- Guardar el archivo de salida

  Guarda en un archivo los datos visibles en el panel **Resultados**.

- Guardar todos los archivos de salida

  Guarda todos los archivos de salida generados de las asignaciones dinámicas[402]. Consulte también el Tutorial 4[103].

- Volver a generar archivo de salida

  Vuelve a generar los datos visibles en el panel **Resultados**

- Ejecutar script SQL/NoSQL (*ediciones Professional y Enterprise*)

  Si hay un script SQL/NoSQL en el panel **Resultados**, el script ejecuta la asignación en la base de datos de destino, teniendo en cuenta las acciones de tabla definidas.

- Insertar/Quitar marcador

  Inserta o quita un marcador en la posición del cursor en el panel **Resultados**.

- Marcador anterior/siguiente

  Navega hasta el marcador anterior/siguiente en el panel **Resultados**.

- Quitar todos los marcadores

  Quita todos los marcadores que están definidos en el panel **Resultados**.

- Texto XML en pretty-print

  Ajusta el formato del documento XML en el panel **Resultados** para presentarlo de forma estructurada. A cada nodo secundario se le aplica una sangría (1 tabulación) con respecto a su primario. Para controlar el tamaño de la sangría use el cuadro de diálogo Características de la vista Texto[68] del grupo de opciones *Tabulaciones*.

- Configurar la vista Texto

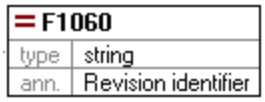
Abre el cuadro de diálogo Configurar la vista Texto, donde puede personalizar la configuración de esta vista en los paneles **XQuery** (ediciones *Professional y Enterprise*), **Resultados** y **XSLT**. También muestra las teclas de acceso rápido que se pueden usar en la ventana. Para más información consulte el apartado Características de la vista Texto 68.

11.8 Vista

En este apartado puede consultar todos los comandos del menú **Vista**.

- Mostrar anotaciones

  Muestra las anotaciones del esquema XML en la ventana del componente. También puede activar esta opción con el botón de la barra de herramientas  . Si también se activa el icono **Mostrar tipos**, el tipo y la anotación aparecen en una tabla (*imagen siguiente*). También puede usar las anotaciones para etiquetar conexiones. Para más detalle consulte Configuración de la conexión 58.

- Mostrar tipos

  Muestra los tipos de datos del esquema para cada elemento o atributo. También puede activar esta opción con el botón de la barra de herramientas  . Si también se activa el icono **Mostrar anotaciones**, el tipo y la anotación aparecen en una tabla (*véase el punto anterior*).

- Mostrar biblioteca en el título de la función

  Muestra el nombre de la biblioteca en paréntesis en el título de la función. También puede activar esta opción con el botón de la barra de herramientas  .

- Mostrar información rápida

  Muestra información rápida con un texto explicativo al pasar el puntero del ratón por encima de la función. Si también se activa el icono **Mostrar información rápida**, también puede ver la información sobre los tipos de datos en el componente.

- Opciones de visualización XBRL (*Enterprise Edition*)

  MapForce permite configurar estas opciones XBRL:

  - El lenguaje de etiquetado de los elementos XBRL y sus anotaciones
  - Los roles de etiquetado preferidos para nombres de elementos XBRL
  - El tipo específico de roles de etiquetado de las anotaciones de los elementos XBRL
  - Paquetes de taxonomías XBRL personales

- Mostrar los conectores del componente seleccionado/Mostrar los conectores desde su origen a su destino

  Permite elegir qué conexiones resaltar. Para saber más acerca de cómo funciona esta opción consulte Conexiones 48.

- Zoom

  Abre el cuadro de diálogo de **zoom**, donde puede indicar el factor de zoom en formato numérico o con un control deslizante.

- Atrás/Adelante

  Retrocede o avanza en las asignaciones que están abiertas en el panel Asignación.

- Barra de estado

  Muestra/oculta la **barra de estado** que está visible bajo la ventana **Mensajes**.

- Bibliotecas/Administrar bibliotecas

  Muestra/oculta la ventana **Biblioteca**, que contiene todas las funciones. La opción **Administrar bibliotecas** activa o desactiva la ventana **Administrar bibliotecas**.

- Mensajes

  Muestra/oculta la ventana Mensajes [25]. La ventana **Mensajes** se activa automáticamente cuando se genera código para mostrar el resultado de la validación.

- Vista general

  Muestra/oculta la ventana Vista general [24]. Arrastre el rectángulo rojo por la ventana de vista general para navegar por el panel **Asignación**.

- Ventana del proyecto (*ediciones Professional y Enterprise*)

  Muestra/oculta la ventana **Proyecto**.

- Ventanas del depurador (*ediciones Professional y Enterprise*)

  El modo depurador permite analizar el contexto en que se produce un valor en concreto. Puede encontrar esta información directamente en la asignación y en los las ventanas **Valores**, **Contexto** y **Puntos de interrupción**.

11.9 Herramientas

En este apartado puede consultar todos los comandos del menú **Herramientas**.

- Recursos globales

  Abre el cuadro de diálogo **Administrar recursos globales** donde puede agregar, editar o eliminar configuraciones que son aplicables en varias aplicaciones de Altova. Para más información, consulte Recursos globales de Altova[432].

- Configuración activa

  Permite seleccionar la configuración de recurso global que está activa en una lista que enumera todas las configuraciones definidas previamente. Para crear y configurar diferentes tipos de recursos globales, consulte Recursos globales de Altova[432].

- Crear asignación invertida

  Crea una asignación invertida a partir de la asignación activa que será la base de una asignación nueva. Esto significa que el componente de origen pasa a ser el componente de destino y viceversa. Tenga en cuenta que en la asignación invertida solamente se conservan las conexiones directas entre los componentes de la asignación original. Es muy posible que la asignación resultante no sea válida o que no se pueda ejecutar (al abrir el panel **Resultados**) sin antes realizar ciertos cambios. Por lo tanto, puede que tenga que editar a mano la asignación nueva.

  Estos son los datos que se conservan:

  - Las conexiones directas entre los componentes
  - Las conexiones directas entre los componentes en una asignación encadenada
  - El tipo de conexión[50]: estándar, mixto, copia total
  - La configuración de componentes de paso a través
  - Los componentes de BD (*en las ediciones Professional y Enterprise*)

  Estos son los datos que *no* se conservan:

  - Las conexiones a través de funciones, filtros, etc.
  - Las funciones definidas por el usuario
  - Los componentes de servicio web (*Enterprise Edition*)

- Gestor de taxonomías XBRL (*Enterprise Edition*)

  El gestor de taxonomías XBRL es una herramienta que permite instalar y administrar taxonomías XBRL.

- Gestor de esquemas XML

  El Gestor de esquemas XML es una herramienta de Altova que ofrece una forma centralizada de instalar y gestionar esquemas XML (DTDs para XML y Esquemas XML) para usarlos en todas las aplicaciones de Altova compatibles con XBRL. Para más información, consulte el Gestor de esquemas[130].

- Personalizar

  Esta opción permite personalizar la interfaz gráfica del usuario de MapForce, incluidas las barras de herramientas que aparecen en la interfaz. También permite personalizar los menús[468] y las teclas de

acceso rápido[469].

- Restaurar barras de herramientas y ventanas

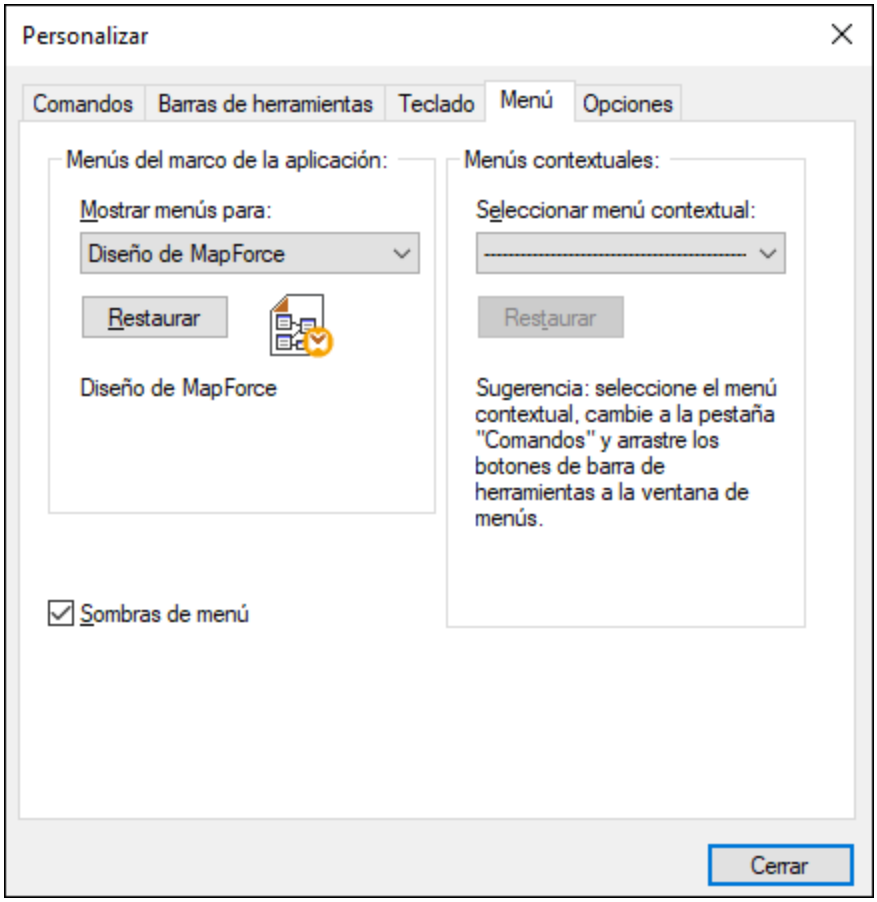
  Restaura las barras de herramientas, las ventanas de ayuda de entrada, las ventanas acopladas, etc. a sus valores predeterminados. Es necesario reiniciar MapForce para que los cambios surtan efecto.

- Opciones

  Al hacer clic, se abre el cuadro de diálogo **Opciones** que permite cambiar la configuración predeterminada de MapForce. Para más información consulte Opciones[472].

11.9.1 Personalizar menús

Puede personalizar los menús estándar de MapForce, así como los menús contextuales (p.ej., para añadir, cambiar o eliminar comandos). También puede restaurar cualquier menú personalizado a su estado predeterminado (**Restaurar**). Para personalizar menús vaya al menú **Herramientas**, haga clic en **Opciones** y después en la pestaña **Menú** (*imagen siguiente*).



Personalizar menús
La barra de menú **Predeterminada** es la que aparece si no hay ningún documento abierto en la ventana principal. La barra de menú **Diseño de MapForce** es la que aparece cuando se abren una o más asignaciones. Cada una de estas barras de herramientas se puede personalizar por separado y los cambios que haga en una de ellas no afectarán a las demás.

Para personalizar una barra de herramientas, selecciónela en la lista desplegable **Mostrar menús para**. Ahora cambie a la pestaña *Comandos* y arrastre los comandos que quiera desde la lista de comandos hasta la barra de herramientas o el menú.

Eliminar comandos de menú y restablecer barras de menú
Para eliminar un menú entero o uno de sus componentes:

1. Seleccione una de estas opciones de la lista desplegable **Mostrar menús para**:

   - **Predeterminado** (muestra los menús disponibles si no hay ningún documento abierto)
   - **Diseño de MapForce** (muestra los menús disponibles si hay una asignación abierta)

2. Con el cuadro de diálogo "Personalizar" abierto, seleccione (i) el menú que quiere borrar de la barra de herramientas de la aplicación o (ii) el comando que quiere eliminar de uno de estos menús.
3. Ahora puede (i) arrastrar el menú fuera desde la barra de herramientas o el comando fuera del menú, o (ii) hacer clic con el botón derecho en el menú o el comando de menú y seleccionar **Eliminar**.

Puede restaurar cualquier barra de menú a su estado original de instalación; para ello selecciónela en la lista desplegable **Mostrar menús para** y haga clic en el botón **Restaurar**.

Personalizar los menús contextuales de la aplicación
Los menús contextuales son aquellos que aparecen si hace clic con el botón derecho en ciertos objetos de la interfaz de la aplicación. Para personalizar esos menús contextuales:

1. Seleccione el menú contextual de la lista desplegable **Seleccionar menú contextual**. Se abre el menú contextual.
2. Haga clic en la pestaña **Comandos**.
3. Arrastre un comando desde la lista Comandos hasta el menú contextual.
4. Para eliminar un comando de un menú contextual, haga clic con el botón derecho en ese comando del menú contextual y seleccione **Eliminar**. Otra opción es arrastrar el comando fuera del menú contextual.

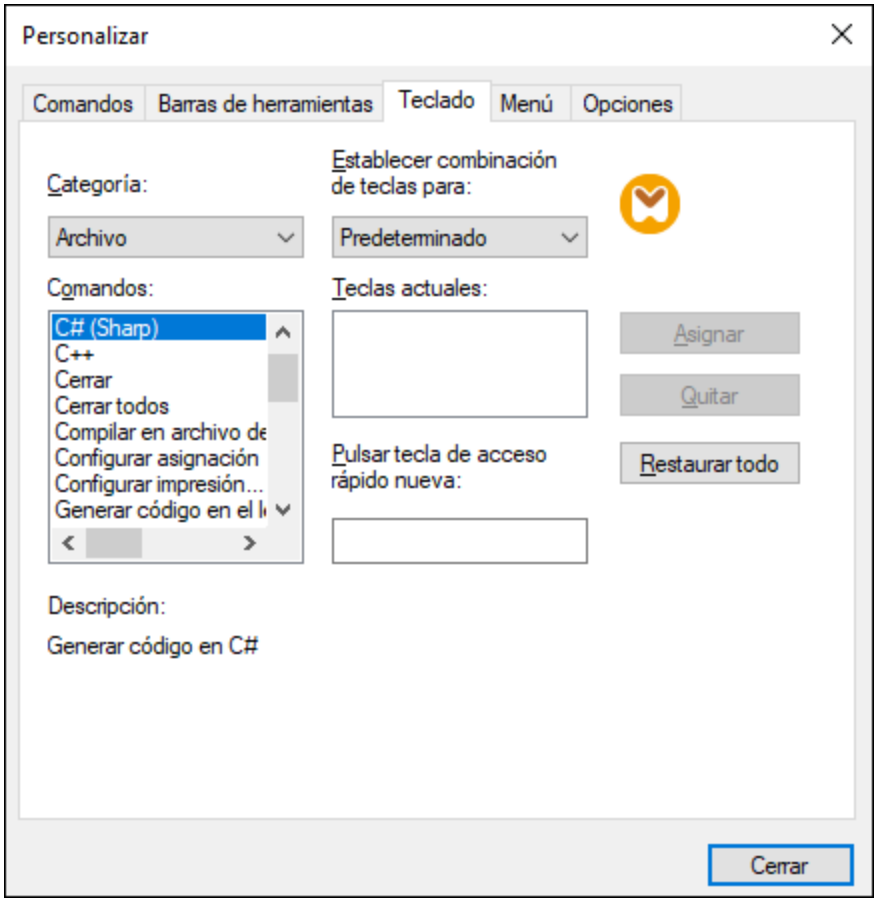
Puede restaurar cualquier menú contextual a su estado original de instalación; para ello selecciónela en la lista desplegable **Seleccionar menú contextual** y haga clic en el botón **Restaurar**.

Sombras de menú
Marque la casilla *Sombras de menú* para que todos los menús tengan sombra.

11.9.2 Personalizar teclas de acceso rápido

En MapForce las teclas de acceso rápido se definen y modifican así: haga clic en el menú **Herramientas** y después en **Personalizar**. Después haga clic en la pestaña **Teclado**. Para asignar una tecla de acceso rápido a un comando:

1. Seleccione **Herramientas | Personalizar** y haga clic en la pestaña *Teclado* del cuadro de diálogo **Personalizar** (*imagen siguiente*).
2. Haga clic en el cuadro combinado *Categoría* y seleccione el menú de MapForce donde está el comando.
3. En el cuadro de lista *Comandos* seleccione el comando al que desea asignar una tecla de acceso rápido.
4. Haga clic en el cuadro de texto *Pulsar tecla de acceso rápido nueva* y haga clic en el botón **Asignar** para asignar la tecla de acceso rápido al comando.



La tecla de acceso rápido aparece ahora en el cuadro de lista *Teclas actuales*. Para borrar el texto del cuadro de texto *Pulsar tecla de acceso rápido,* pulse cualquier tecla de control (**Ctrl**, **Alt** o **Mayús**). Para eliminar una combinación de teclas de la lista *Teclas actuales*, haga clic en ella y pulse **Quitar**.

Nota: El cuadro combinado *Establecer combinación de teclas para:* no tiene por ahora ninguna función.

Teclas de acceso rápido

Estas son las teclas de acceso rápido asignadas por defecto:

F1	Menú Ayuda
F2	Marcador siguiente (panel *Resultados*)
F3	Buscar Siguiente
F10	Activar barra de menú
+ del teclado numérico	Expandir el nodo actual

- del teclado numérico	Contraer el nodo actual
* del teclado numérico	Expandir todo a partir del nodo actual
CTRL + TAB	Cambiar de una asignación a otra
CTRL + F6	Recorrer todas las ventanas abiertas
CTRL + F4	Cerrar la asignación activa
Alt + F4	Cerrar MapForce
Alt + F, F, 1	Abrir el último archivo
Alt + F, T, 1	Abrir el último proyecto
CTRL + N	Archivo nuevo
CTRL + O	Abrir archivo
CTRL + S	Guardar archivo
CTRL + P	Imprimir archivo
CTRL + A	Seleccionar todo
CTRL + X	Cortar
CTRL + C	Copiar
CTRL + V	Pegar
CTRL + Z	Deshacer
CTRL + Y	Rehacer
Supr	Eliminar componente (con previo aviso)
Mayús + Supr	Eliminar componente (sin previo aviso)
CTRL + F	Buscar
F3	Buscar siguiente
Mayús + F3	Buscar anterior

Teclas de dirección

(arriba / abajo)	Seleccionar el siguiente/anterior elemento del componente
Esc	Abandonar los cambios/cerrar el cuadro de diálogo
Entrar	Confirmar una selección

Teclas del panel Resultados

CTRL + F2	Insertar/Quitar marcador
F2	Siguiente marcador
Mayús + F2	Marcador anterior
CTRL + Mayús + F2	Quitar todos los marcadores

Teclas de zoom

CTRL + rueda del ratón hacia adelante	Acercarse
CTRL + rueda del ratón hacia detrás	Alejarse
CTRL + 0 (cero)	Restablecer el nivel de zoom

11.9.3 Opciones

Puede cambiar las opciones generales y preferencias de MapForce con el comando **Herramientas | Opciones**. A continuación puede ver las opciones disponibles.

- Generales

  En la sección *Generales* puede definir estas opciones:

  - *Mostrar logotipo | Al iniciar el programa:* Muestra u oculta el logotipo (pantalla de bienvenida) cuando se inicia MapForce.

  - En la sección *Visualización de asignaciones* puede configurar estas opciones:

    - Habilita y deshabilita el fondo degradado del panel Asignación (*Mostrar fondo degradado*).
    - Puede limitar la visualización de anotaciones a N líneas (*Limitar la presentación de anotaciones a*) Por ejemplo, si ha configurado esta opción con el valor 2 y el texto de la anotación tiene 3 líneas, en la asignación solamente se verán las dos primeras líneas. Esta opción también afecta a las instrucciones SELECT que estén visibles en los componentes.
    - También puede limitar la visualización de comentarios de componentes [32] a N líneas (*Limitar la visualización de comentarios de componentes a N líneas*) Por ejemplo, si ha limitado la visualización de comentarios a una línea y un comentario contiene más de una línea, en la caja sólo se verá la primera línea. Si configura la propiedad con el valor `0` no se verá ningún comentario. Recuerde que la opción *Limitar la visualización de comentarios* no afecta a los comentarios de los componentes [34].

  - *Codificación predeterminada para componentes nuevos.*

    Nombre de la codificación: La codificación predeterminada para archivos XML nuevos se puede definir seleccionando la opción que prefiera de la lista desplegable. Si selecciona una codificación de dos o cuatro bytes como codificación predeterminada (es decir, UTF-16, UCS-2 o UCS-4) también puede elegir entre el orden de bytes little-endian o big-endian. Esta configuración se puede cambiar individualmente para cada uno de los componentes (véase Cambiar la configuración de los componentes [39]).

    Orden de bytes: Los documentos con codificación de caracteres de dos o cuatro bytes se pueden guardar con criterio de ordenación de bytes little-endian o big-endian. También puede indicar si se debe incluir una marca de orden de bytes.

  - *Configurar la vista previa:* La opción *Usar tiempo de espera de la ejecución* indica el tiempo de espera de ejecución para la vista previa de resultados en el panel **Resultados**.

  - *Al activar el panel Resultados:* Puede generar resultados en archivos temporales o escribir el resultado directamente en un archivo de resultados (*imagen siguiente*).

    Generar resultados en archivos temporales: Esta es la opción predeterminada. Si el archivo de salida contiene carpetas que no existen todavía, Mapforce se encargará de crearlas. En las ediciones Professional y Enterprise: Si quiere implementar y ejecutar la asignación de datos en un servidor, todos los directorios de la ruta deben existir también en el servidor; de lo contrario se generará un error de ejecución.

Escribir directamente en archivos de salida finales Si la ruta de salida contiene carpetas que no existen todavía se generará un error de asignación. Esta opción sobrescribe los archivos de salida sin solicitar la confirmación previa del usuario.

- *Presentar texto en tramos de N millones de caracteres* Indica el tamaño máximo del texto que aparece en el panel **Resultados** cuando se genera la vista previa de resultados de asignaciones que generan archivos XML. Si el resultado excede este valor, el texto restante se hará visible al hacer clic sobre un botón **Cargar más**. Para más información, consulte Vista previa de resultados [64].

Edición

Estas son las opciones disponibles en esta página:

- *Alinear componentes al arrastrarlos con el ratón:* Indica si los componentes y funciones deben alinearse con los demás componentes cuando se arrastran con el ratón (véase Alinear componentes [40]).

- *Eliminación inteligente de componentes:* MapForce permite conservar conexiones aunque se hayan eliminado algunos componentes de transformación [32]. Mantener conexiones puede ser especialmente útil si tiene varias conexiones secundarias, ya que no tiene que restaurar cada una de ellas manualmente después de eliminar el componente de transformación en cuestión. Para más detalles consulte Conservar conexiones tras eliminación de componentes [61].

Mensajes

En la ventana *Mensajes* puede volver a habilitar notificaciones de mensajes deshabilitados previamente al activar la casilla No volver a mostrar este mensaje.

Generación (*ediciones Enterprise y Professional*)

En *Generación de código* puede configurar opciones relacionadas con la generación de código de programa y de archivos de ejecución de MapForce Server.

Java

Puede que necesite añadir esta ruta personal de acceso a un equipo virtual java si está usando un equipo virtual java que no tiene instalador ni crea entradas de registro (por ejemplo, OpenJDK, de Oracle). También puede querer usar esta ruta para suprimir, por la razón que fuere, cualquier otra ruta que MapForce haya detectado automáticamente. Para más detalles consulte Java [475].

XBRL (*Enterprise Edition*)

En MapForce puede configurar estas opciones XBRL generales para toda la aplicación:

- El idioma de etiqueta de los elementos XBRL y de sus anotaciones
- Las funciones de etiqueta preferidas para el nombre de los elementos XBRL
- El tipo concreto de funciones de etiqueta para las anotaciones de los elementos XBRL
- Paquetes de taxonomías XBRL personales

Depurador (*ediciones Enterprise y Professional*)

En la sección *Depurador* puede definir estas opciones:

- *Longitud máxima de almacenamiento de valores:* Define la longitud de cadena de los valores que aparecen en la ventana **Valores** (15 caracteres como mínimo). Recuerde que si define una longitud de almacenamiento alta, puede que se agote la memoria del sistema.

- *Guardar historial de seguimiento completo* Determina si MapForce debe guardar el historial de todos los valores procesados por todos los conectores de todos los componentes de la asignación mientras dure la depuración. Si activa esta opción, todos los valores procesados por MapForce desde el principio de la ejecución de depuración se almacenarán en memoria y se podrán analizar en la ventana **Valores** hasta que se detenga la depuración. No se recomienda activar esta opción si tiene pensado depurar asignaciones que hagan un uso intensivo de datos porque podría ralentizar la ejecución de depuración y agotar la memoria del sistema. Si desactiva esta opción, MapForce guardará solamente el historial de seguimiento más reciente de los nodos relacionados con la posición actual de la ejecución.

Base de datos (*ediciones Enterprise y Professional*)

En la sección *Base de datos* puede definir la consulta de la base de datos.

Proxy de red

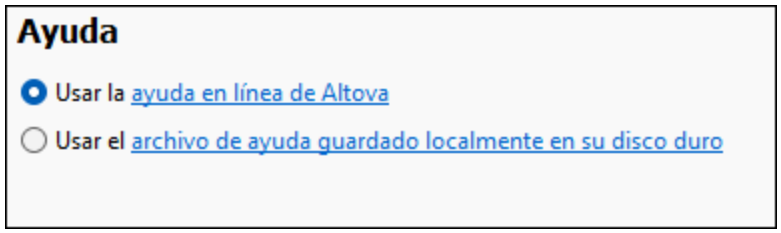
La sección *Proxy de red* permite personalizar la configuración del proxy de red. Esta opción afecta a cómo se conecta la aplicación a Internet. Por defecto, la aplicación usa la configuración del proxy del sistema, por lo que no necesita configurarlo para que funcione. Para más detalles consulte Configuración del proxy de red 477.

Ayuda

MapForce contiene la ayuda (el manual del usuario) en dos formatos:

- La ayuda en línea, en formato HTML, que puede encontrar en el sitio web de Altova. Para acceder a la ayuda en línea necesita tener acceso a Internet.
- Un archivo PDF de ayuda que se instala en el equipo al instalar MapForce. La versión local es un PDF llamado `MapForce.pdf` que puede encontrar en la carpeta de la aplicación (en el directorio Archivos de programa). Si no tiene acceso a Internet siempre puede abrir el archivo local de ayuda.

La opción Ayuda (*imagen siguiente*) permite seleccionar cuál de los dos formatos se abre al hacer clic en el comando **Ayuda (F1)** del menú **Ayuda**.

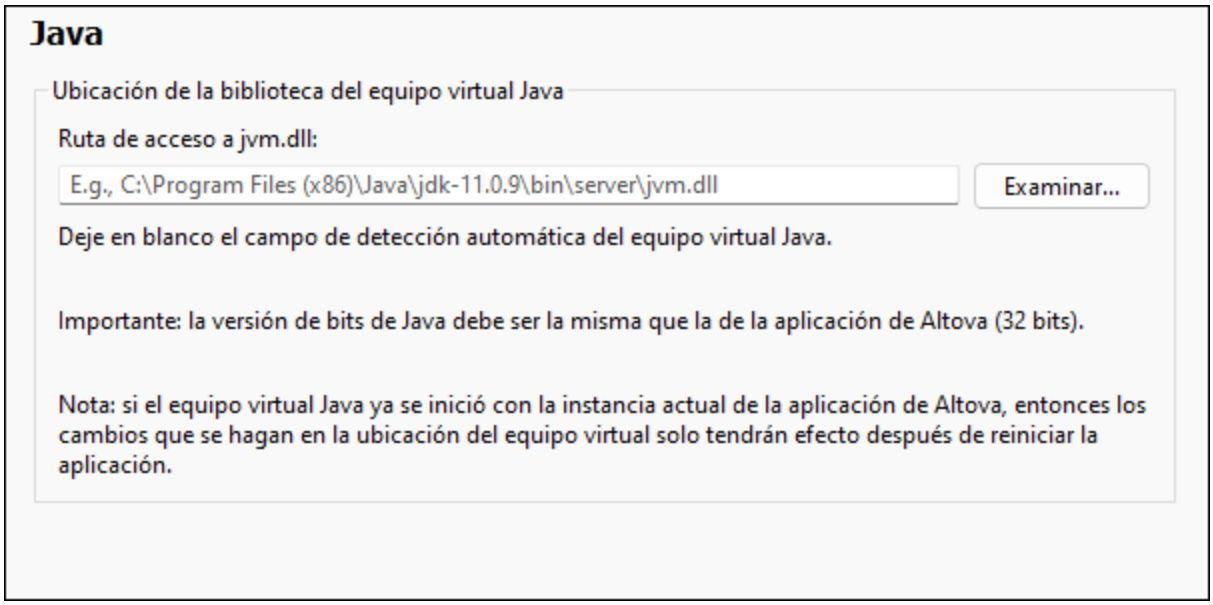


Puede cambiar esta opción en cualquier momento. Los enlaces de esta sección (*imagen anterior*) abren la ayuda en el formato que haya seleccionado.

11.9.3.1 Java

En la pestaña *Java* puede introducir la ruta de acceso a un equipo virtual java en su sistema de archivos. Tenga en cuenta que no siempre es necesario agregar una ruta de acceso personal a un equipo virtual. Por defecto, MapForce intenta detectar esta ruta automáticamente leyendo (en este orden) el registro de Windows y la variable de entorno JAVA_HOME. Si se detecta automáticamente cualquier otra ruta de equipo virtual java, tendrá prioridad la ruta personal que se indica en este cuadro de diálogo.

Puede que necesite añadir esta ruta personal de acceso a un equipo virtual java si está usando un equipo virtual java que no tiene instalador ni crea entradas de registro (por ejemplo, OpenJDK, de Oracle). También puede querer usar esta ruta para suprimir, por la razón que fuere, cualquier otra ruta que MapForce haya detectado automáticamente.



Observe lo siguiente:

- la ruta de acceso al equipo virtual java es común a todas las aplicaciones de escritorio de Altova (no a las de servidor). En consecuencia, si cambia esta ruta en una de ellas, el cambio afectará automáticamente al resto de aplicaciones de Altova.
- la ruta debe apuntar al archivo jvm.dll desde los directorios **\bin\server** o **\bin\client**, relativos al directorio en el que está instalado el JDK.
- la plataforma de MapForce (versión de 31 o de 64 bits) debe ser la misma que la del JDK.
- después de cambiar la ruta de acceso al escritorio virtual java debe reiniciar MapForce para que surta efecto la nueva configuración.

11.9.3.2 Opciones de red

La sección **Opciones de red** (*imagen siguiente*) permite configurar las opciones de red.

Direcciones IP
Cuando los nombres de host se resuelven en más de una dirección en redes mixtas IPv4/IPv6, marcar esta casilla indica que se deben usar las direcciones IPv6. Si no se marca esta casilla en dichos entornos y hay direcciones IPv4 disponibles, se usan direcciones IPv4.

Tiempo de espera

- *Tiempo de espera de transferencia:* Si se alcanza este límite al transferir dos paquetes de datos consecutivos (enviados o recibidos), se anula la transferencia al completo. Puede indicar los valores en segundos [s] o milisegundos [ms]; el valor predeterminado son 40 segundos. Si no se marca esta opción no existe ningún límite de tiempo para anular la transferencia.
- *Tiempo de espera de conexión:* Este es el límite de tiempo en el que debe establecerse la conexión, incluido el tiempo que se tarda en establecer la comunicación (handshake). Puede indicar los valores en segundos [s] o milisegundos [ms]; el valor predeterminado son 300 segundos. Este tiempo de espera no se puede deshabilitar.

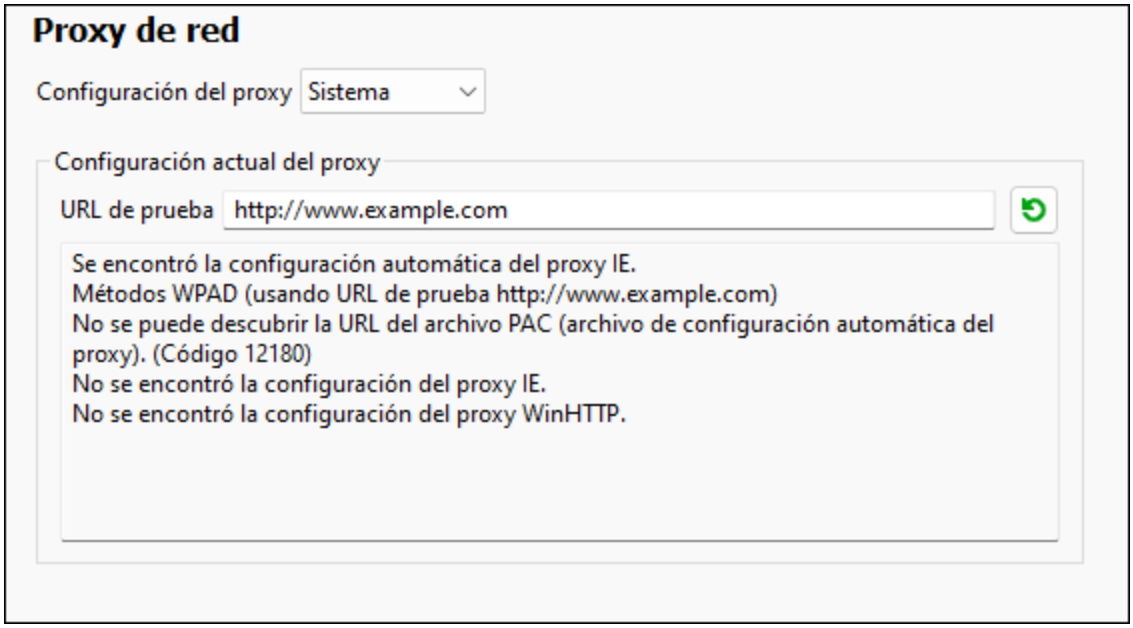
Certificado

- *Verificar el certificado del servidor TLS/SSL:* Si se marca esta opción se comprueba la autenticidad del certificado del servidor, para lo que se comprueba la cadena de firmas digitales hasta que se alcanza un certificado raíz de confianza. Esta opción está marcada por defecto. Si no se marca esta opción, la comunicación no es segura y no se detecta si hay ataques (por ejemplo, ataques de suplantación de identidad). Tenga en cuenta que esta opción no comprueba si el certificado pertenece al servidor con el que se está comunicando. Para habilitar la seguridad al completo debe marcar las casillas de certificado y de identidad (*véase la opción siguiente*).
- *Verificar la identidad del servidor TLS/SSL:* Si marca esta opción, se comprueba si el certificado pertenece al servidor con el que se quiere establecer la conexión. Para ello se comprueba si el nombre del servidor de la URL es el mismo que el del certificado. Esta opción está marcada por defecto. Si no se marca esta opción, no se comprueba la identidad del servidor. Recuerde que esta opción no habilita la verificación del certificado del servidor. Para habilitar la seguridad al completo debe marcar tanto la casilla de certificado como la de identidad (*véase la opción anterior*).

11.9.3.3 Proxy de red

El cuadro de diálogo *Proxy de red* permite personalizar la configuración del proxy de red. Esta configuración afecta a cómo la aplicación se conecta a Internet (p.ej. ara validar un documento XML). El sistema viene con una configuración predeterminada para el proxy, por lo que este funcionará sin necesidad de configurarlo, pero si quiere usar un proxy de red alternativo puede usar estas opciones para cambiar la configuración como quiera.

Nota: La configuración del proxy de red es común a todas las aplicaciones de Altova MissionKit. En consecuencia, si cambia esta configuración en cualquiera de esas aplicaciones, el cambio afectará automáticamente a todas las demás.



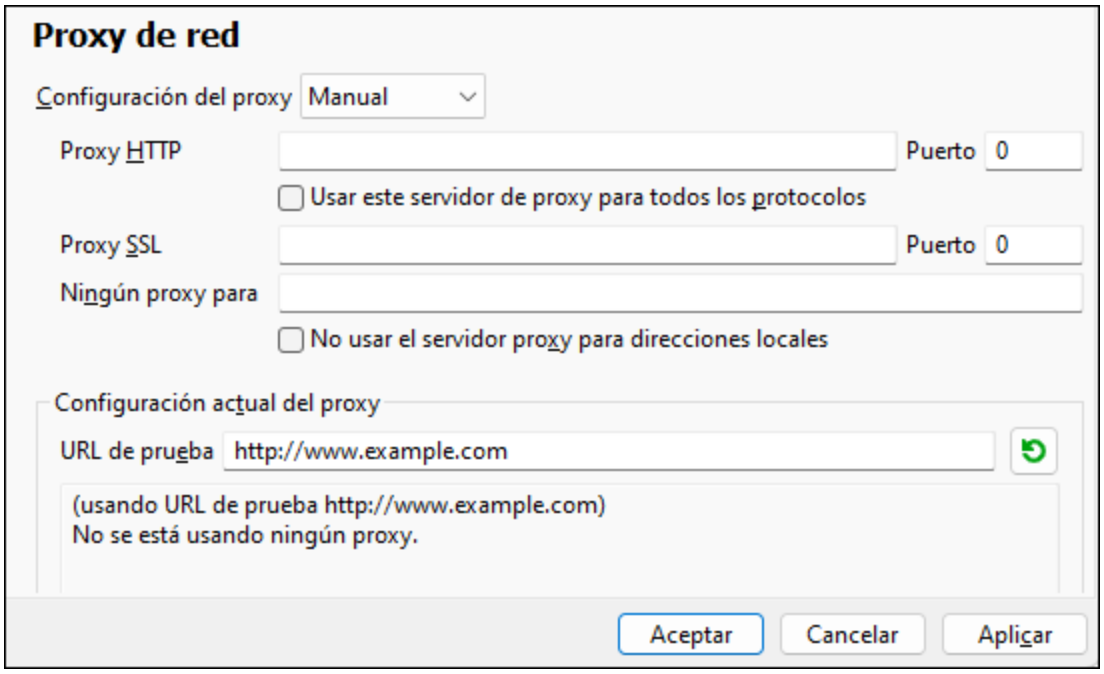
Usar la configuración del proxy del sistema
Usa los parámetros de Internet Explorer (IE), que se pueden configurar desde las opciones del proxy de red. También consulta los parámetros configurados con **`netsh.exe winhttp`**.

Configuración automática del proxy
Existen las siguientes opciones:

- *Configuración de detección automática:* consulta un script WPAD (**`http://wpad.LOCALDOMAIN/wpad.dat`**) vía DHCP o DNS y lo usa para configurar el proxy.
- *URL del script*: indica una HTTP URL a un script (`.pac`) de configuración automática del proxy cuyos parámetros se aplican para configurar el proxy.
- *Volver a cargar*: reinicia y vuelve a cargar la configuración automática actual del proxy. Esta acción requiere Windows 8 o superior y puede llegar a tardar 30 segundos en tener efecto.

Configuración manual del proxy
Puede indicar manualmente el nombre completo de host y el puerto para los proxys de los respectivos

protocolos. Es posible que haya un esquema compatible incluido en el nombre de host (por ejemplo: **`http://hostname`**). Si el proxy es compatible no es necesario que el esquema sea el mismo que el protocolo correspondiente.



Existen las siguientes opciones:

- *Proxy HTTP:* usa el nombre de host y puerto especificados o el protocolo HTTP. Si selecciona *Usar este servidor de proxy para todos los protocolos* se usan el nombre de host y el puerto del Proxy HTTP para todos los protocolos.
- Proxy SSL: usa el nombre de host y puerto especificados para el protocolo SSL.
- *Ningún proxy para:* muestra una lista de elementos separados por punto y coma (**`;`**) que pueden ser nombres de host, nombres de dominios o direcciones IP para hosts para los que no hay que usar proxy. Las direcciones IP no se pueden truncar y las direcciones IPv6 deben colocarse entre corchetes (por ejemplo: **`[2606:2800:220:1:248:1893:25c8:1946]`**). Los nombres de dominio deben empezar por punto (por ejemplo: **`.example.com`**).
- *No use el servidor proxy para direcciones locales*: si se marca esta opción, se añade el elemento **`<local>`** a la lista *Ningún proxy para*. Si se selecciona esta opción no se usará proxy para: (i) **`127.0.0.1`**, (ii) [::1], (iii) todos los nombres de host que no contengan punto (**`.`**).

Nota: Si ha configurado un servidor de proxy y quiere implementar una asignación en Altova FlowForce Server, debe seleccionar la opción *No usar el servidor proxy para direcciones locales*.

Configuración actual del proxy
Proporciona un registro detallado de la detección del proxy. Se puede actualizar con el botón **Actualizar** a la derecha de *URL de prueba* (por ejemplo, al cambiar la URL de prueba o cuando se ha cambiado la configuración del proxy).

- *URL de prueba:* una URL de prueba se puede usar para ver qué proxy hay que usar para esa URL en concreto. No se trata de una URL de entrada/salida. Este campo no debe estar vacío si se ha optado por la configuración automática del proxy (seleccionando *Usar la configuración del proxy del sistema* o

Configuración automática del proxy).

11.10 Ventanas

En este apartado puede consultar todos los comandos del menú **Ventanas**.

- En cascada

  Este comando reorganiza todas las ventanas abiertas **en forma de cascada** (es decir, escalonadas).

- En mosaico horizontal

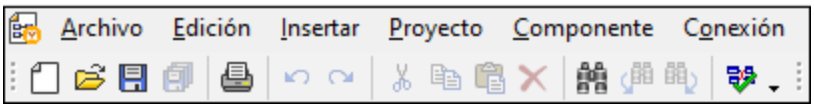
  Este comando reorganiza todas las ventanas abiertas **en forma de mosaico horizontal**, mostrando todas las ventanas a la vez.

- En mosaico vertical

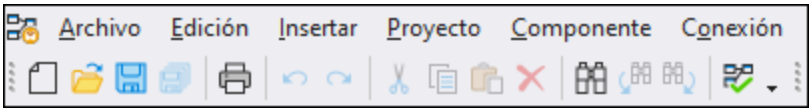
  Este comando reorganiza todas las ventanas abiertas **en forma de mosaico vertical**, mostrando todas las ventanas a la vez.

- Tema clásico/claro/oscuro

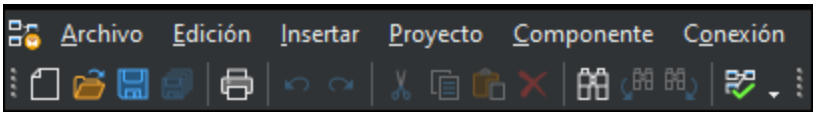
  MapForce permite elegir entre estos temas: *clásico*, *claro* y *oscuro*. A continuación puede ver ejemplos de los tres temas. La opción predeterminada es el tema clásico.



Tema clásico



Tema claro



Tema oscuro

- 1 <NombreAsignación>

  Hace referencia al primer diseño de asignación abierto. Si hay más asignaciones abiertas en ese momento también aparecerán en el menú contextual.

- Ventanas

  Esta lista enumera todas las ventanas que están abiertas en cada momento y permite cambiar de una ventana a otra. También puede usar las teclas **Ctrl+Tabulador** o **Ctrl+F6** para pasar de una ventana a otra.

11.11 Ayuda

En este apartado puede consultar todos los comandos del menú **Ayuda**.

- Ayuda (F1)

  El comando **Ayuda (F1)** abre la documentación de ayuda de la aplicación (el manual del usuario). La ayuda que se abre de forma predeterminada es la ayuda en línea en formato HTML.

  Si no tiene acceso a Internet o por algún motivo no quiere usar la ayuda en línea, siempre puede usar la versión local del manual del usuario. La versión local es un PDF llamado `MapForce.pdf` que puede encontrar en la carpeta de la aplicación (en el directorio Archivos de programa).

  Si quiere cambiar el formato predeterminado (ayuda en línea o PDF local) puede hacerlo en la sección Ayuda del cuadro de diálogo Opciones (comando de menú **Herramientas | Opciones**).

- Activación del software

  Asignar una licencia al producto
  Tras descargar el producto de software de Altova puede registrarlo o activarlo con una clave de evaluación gratuita o con una clave de licencia permanente.

  - ***Licencia de evaluación gratuita***. Cuando inicie el software por primera vez, tras haberlo descargado e instalado, aparecerá el cuadro de diálogo **Activación del software**. Este cuadro de diálogo incluye un botón para solicitar una licencia de evaluación gratuita. Haga clic en este botón para obtener su licencia. Al hacer clic en este botón, se generará un código hash para el ID de su equipo que se enviará a Altova a través de HTTPS. La información de la licencia se devolverá al equipo a través de una respuesta HTTP. Una vez la licencia se haya creado con éxito, aparecerá un cuadro de diálogo al respecto en su aplicación de Altova. Al hacer clic en **Aceptar** en este cuadro de diálogo se activará el software durante 30 días **en ese equipo particular**.

  - ***Clave de licencia permanente***. El cuadro de diálogo **Activación del software** también incluye un botón para comprar una clave de licencia permanente. Este botón conduce a la tienda en línea de Altova, donde podrá adquirir una clave de licencia permanente para el producto. Recibirá por correo electrónico un archivo que contiene sus datos de la licencia.

    Existen tres tipos de licencias permanentes: de tipo *instalado*, *de usuario concurrente* y *de usuario designado*. Las licencias de tipo instalado son cada una para un único equipo. Si adquiere una licencia instalada para *N* ordenadores, la licencia permite utilizar el software hasta en esta cantidad de ordenadores. De la misma manera, una licencia de usuario concurrente para *N* usuarios concurrentes permite a *N* usuarios ejecutar el software de forma concurrente. (El software puede instalarse en 10N ordenadores). Las licencias de usuario designado autorizan a un usuario específico a usar el software en un máximo de 5 equipos distintos. Para activar su software haga clic en **Cargar una licencia nueva** e introduzca la ruta de acceso al archivo de licencia en el cuadro de diálogo que aparece. Por último, haga clic en **Aceptar**.

  Nota: En el caso de licencias para varios usuarios, se le pedirá a cada usuario que introduzca su nombre.

Claves por correo electrónico y las distintas formas de activar las licencias de los productos de Altova
El correo electrónico que recibirá de Altova contiene, en un adjunto, el archivo de la licencia. El archivo de la licencia tiene la extensión **.altova_licenses**.

Para activar su producto de Altova, puede optar por una de las siguientes opciones:

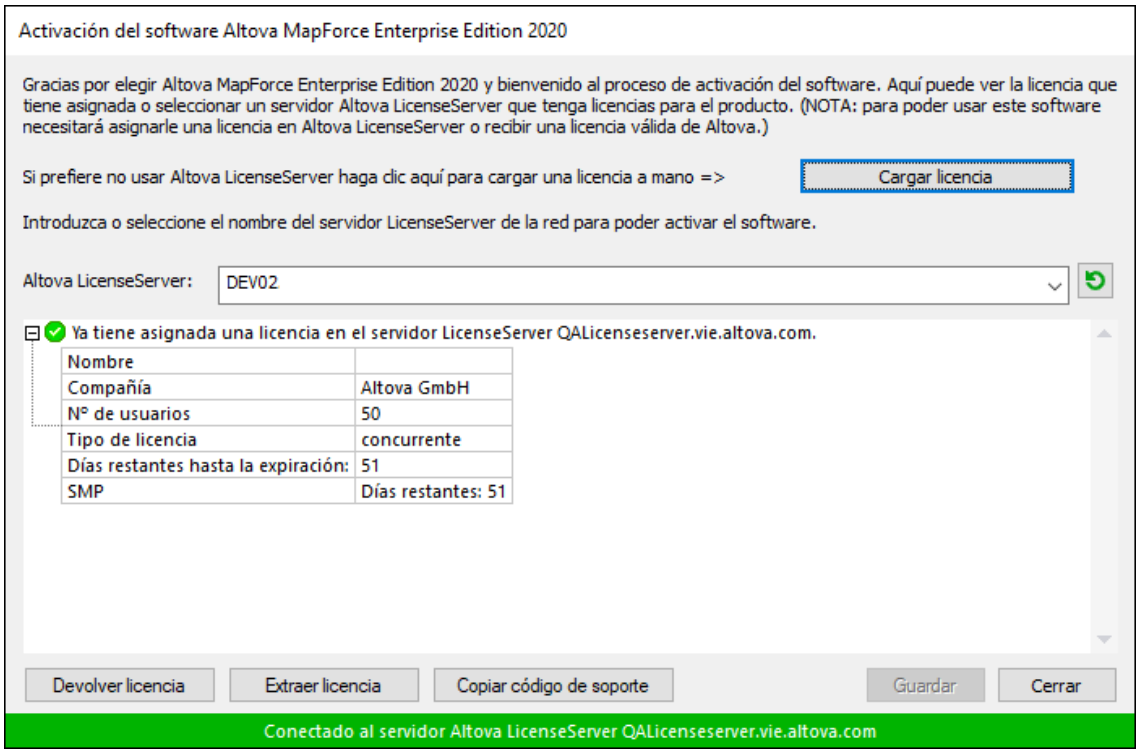
- Guardar el archivo de licencia (**.altova_licenses**) en su equipo, hacer doble clic en el archivo de licencia, introducir los detalles necesarios en el cuadro de diálogo que aparece y finalmente hacer clic en **Aplicar claves**.
- Guardar el archivo de licencia (**.altova_licenses**) en su equipo. En su producto de Altova seleccione el comando de menú **Ayuda | Activación del software** y después **Cargar una licencia nueva**. Puede escribir la ruta de acceso o navegar hasta el archivo de licencia, y luego hacer clic en **Aceptar**.
- Guardar el archivo de licencia (**.altova_licenses**) en su equipo y cargarlo desde esa ubicación a su Altova LicenseServer. Puede elegir entre estas dos opciones: (i) adquirir la licencia de su producto Altova con el cuadro de diálogo de activación de software del producto (*véase más abajo*) o (ii) asignar la licencia al producto de Altova LicenseServer. *Para obtener más información sobre la gestión de licencias con el LicenseServer, lea el resto de esta sección.*

El cuadro de diálogo **Activación del software** (*imagen siguiente*) se abre con el comando **Ayuda | Activación del software**.

Activar el software
Puede activar el software registrando la licencia en el cuadro de diálogo "Activación del software" o asignando una licencia a través de Altova LicenseServer (*ver detalles más abajo*).

- *Registrando la licencia en el cuadro de diálogo "Activación del software"*. En el cuadro de diálogo, haga clic en **Cargar una licencia nueva** y navegue hasta el archivo de la licencia. Haga clic en **Aceptar** para confirmar la ruta de acceso al archivo de licencia y para confirmar los datos que haya introducido (su nombre, en el caso de licencias para más de un usuario). A continuación, haga clic en **Guardar** para finalizar el proceso.

- *Asignando una licencia a través de un servidor Altova LicenseServer de la red:* Para adquirir una licencia a través de un servidor Altova LicenseServer de la red haga clic en el botón **Usar Altova LicenseServer**, situado al final del cuadro de diálogo **Activación del software**. Seleccione el equipo en el que está instalado el LicenseServer que quiere usar. Tenga en cuenta que la autodetección de los License Servers funciona con emisiones enviadas por LAN. Este tipo de emisiones se limitan a una subred, por lo que Altova License Server debe estar en la misma subred que el equipo del cliente para que funcione la autodetección. Si esta no funciona, introduzca el nombre del servidor. Para ello es necesario que el servidor LicenseServer tenga una licencia para su producto en el repositorio de licencias. Si así es, el cuadro de diálogo **Activación del software** emite un mensaje a tal efecto (*ver imagen siguiente donde figura el cuadro de diálogo en Altova XMLSpy*). Haga clic en el botón **Guardar** para adquirir la licencia.



Una vez se ha adquirido una licencia para un equipo específico (es decir, "instalada") del servidor LicenseServer, no se puede devolver al mismo hasta 7 días después. Transcurridos estos 7 días podrá devolver la licencia de ese equipo (con el botón **Devolver licencia**) para que pueda ser adquirida por otro cliente. No obstante, el administrador de LicenseServer puede anular asignaciones de licencias desde la interfaz web del servidor LicenseServer en cualquier momento. Observe que únicamente se pueden devolver las licencias instaladas en equipos específicos, no las licencias concurrentes.

Extracción de licencias
Puede extraer una licencia del repertorio durante un período máximo de 30 días de modo que la licencia se almacene en el equipo donde se ejecuta el producto. Esto le permitirá trabajar sin conexión a Internet, lo cual puede ser útil si desea trabajar en un entorno que no dispone de acceso a su servidor Altova LicenseServer (p. ej. cuando el producto servidor de Altova está instalado en un equipo portátil y el usuario se encuentra de viaje). Mientras la licencia esté extraída, LicenseServer indicará que la licencia está en uso y no podrá ser utilizada por ningún otro equipo. La licencia vuelve de forma automática al de licencias una vez ha finalizado el periodo de extracción. La licencia extraída también se puede insertar en el servidor en cualquier momento con el botón **Insertar** del cuadro de diálogo **Activación del software**.

Siga estas instrucciones para extraer una licencia: (I) En el cuadro de diálogo **Activación del software** haga clic en el botón **Extraer licencia** (*imagen anterior*). (Ii) Aparece el cuadro de diálogo **Extracción de licencias**. Seleccione el periodo de extracción deseado y haga clic en **Extraer**. Así se extraerá la licencia. Ahora, después de haber extraído una licencia, ocurren dos cosas: (i) El cuadro de diálogo **Activación del software** muestra información sobre la extracción de la licencia, incluida la fecha y la hora en la que expira el plazo de extracción y (ii) En lugar del botón **Extraer licencia**, aparece el botón **Insertar licencia**. Para insertar la licencia en cualquier momento dado, basta con hacer clic en este botón. Como la licencia vuelve automáticamente a

su estado de inserción cuando finaliza el plazo de extracción, compruebe que el plazo seleccionado coincide con el período de tiempo que tiene pensado trabajar sin conexión a Internet.

Si la licencia que extrae es una licencia de tipo instalado o una licencia de usuario concurrente, entonces esta se extrae al equipo y está disponible para el usuario que extrajo la licencia. Si la licencia que extrae es una licencia de usuario designado, entonces esta se extrae a la cuenta de Windows del usuario designado. Se pueden extraer licencias en equipos virtuales pero no para escritorios virtuales (en una virtualización de escritorio). Tenga en cuenta que al extraer una licencia de usuario designado, los datos que identifican esa extracción de licencia se almacenan en el perfil del usuario. Para que funcione la extracción de licencias, el perfil del usuario debe estar almacenado en el equipo local que se utilizará para trabajar sin conexión. Si el perfil del usuario se encuentra en una ubicación no local (como un archivo compartido), la extracción se considerará no válida a la hora de iniciar la aplicación de Altova.

Para devolver una licencia esta debe ser de la misma versión principal que el producto de Altova con el que se extrajo. Por tanto, es recomendable devolver la licencia antes de actualizar el producto de Altova correspondiente a la siguiente versión principal.

Nota: Para poder extraer licencias esta característica debe estar habilitada en el servidor LicenseServer. Si esta característica no está habilitada, recibirá un mensaje de error a tal efecto cuando trate de extraer una licencia. Cuando esto ocurra, póngase en contacto con el administrador de su servidor LicenseServer.

Copiar código de soporte
Haga clic en **Copiar código de soporte** para copiar los detalles de la licencia en el portapapeles. Esta es la información que deberá introducir al ponerse en contacto con el equipo de soporte técnico a través del formulario de soporte técnico.

Altova LicenseServer es una práctica herramienta para administrar en tiempo real todas las licencias de Altova de la red y ofrece información detallada sobre cada licencia, asignaciones a clientes y uso de las licencias. La ventaja de usar este producto está en las características administrativas que ofrece para la gestión de grandes volúmenes de licencias de Altova. Altova LicenseServer puede descargarse gratis del sitio web de Altova. Para más información sobre Altova LicenseServer, consulte la documentación de Altova LicenseServer.

Formulario de pedido

Hay dos maneras de comprar licencias para los productos de Altova: con el botón **Comprar una licencia permanente** del cuadro de diálogo **Activación del software** (*ver apartado anterior*) o con el comando **Formulario de pedido**, que le lleva directamente a la tienda en línea de Altova.

Registro del software

Este comando abre la página de registro de productos de Altova en una pestaña del explorador. Si registra el software, recibirá información sobre actualizaciones y versiones nuevas del producto.

Buscar actualizaciones

Comprueba si existe una versión más reciente del producto en el servidor de Altova y emite un mensaje a tal efecto.

Soporte técnico

Es un enlace al centro de soporte técnico de Altova en Internet. El centro de soporte técnico incluye preguntas frecuentes, foros de debate y un formulario para ponerse en contacto con el equipo de soporte técnico de Altova.

- Descargar herramientas gratis y componentes

  Es un enlace al centro de descargas de componentes del sitio web de Altova. Aquí puede descargar una variedad de software adicional para usarlo con los productos de Altova,como procesadores XSLT y XSL-FO y paquetes de integración. Estos componentes suelen ser totalmente gratis.

- MapForce en Internet

  Es un enlace al sitio web de Altova. Aquí encontrará más información sobre MapForce, otros productos de Altova y tecnologías relacionadas.

- Cursos de MapForce

  Es un enlace a la página de cursos del sitio web de Altova. Aquí puede seguir todos los cursos sobre productos y tecnologías relacionados con la línea de software de Altova.

- Acerca de MapForce

  Abre la pantalla de presentación de la aplicación y muestra el número de versión del producto. Si usa la versión de 64 bits de MapForce, esto se ve en el nombre de la aplicación, que lleva el sufijo (x64). La versión de 32 bits no lleva ningún sufijo.

12 Anexos

Estos anexos contienen datos técnicos sobre MapForce e información importante sobre las licencias. También contiene unan lista de términos clave específicos de MapForce y los productos relacionados con MapForce. Esta sección se divide en los apartados siguientes:

12.1 Notas sobre compatibilidad

MapForce es una aplicación de 32/64 bits para Windows compatible con los sistemas operativos siguientes:

- Windows 10, Windows 11
- Windows Server 2016 o superior

Las ediciones Enterprise y Professional son compatibles con plataformas de 64 bits.

12.1.1 Orígenes y destinos de datos compatibles

Al cambiar el lenguaje de transformación de una asignación de MapForce, es posible que algunas funciones no sean compatibles con ese lenguaje en concreto Esta tabla recoge, ordenados por lenguaje, todos los orígenes y destinos de datos compatibles con **MapForce Basic Edition**.

Notas:

- *Built-in* permite ejecutar la asignación haciendo clic en la pestaña **Resultados** de MapForce o bien con MapForce Server.

12.1.2 Funciones compatibles en el código generado

En esta tabla se recogen las funciones relevantes para la generación de código y el alcance de la compatibilidad con **MapForce Basic Edition** para cada lenguaje.

Función	XSLT 1.0	XSLT 2.0	XSLT 3.0
Pasar parámetros a la asignación 147	●	●	●
Suministrar los nombres de los archivos de entrada dinámicamente desde la asignación 402	●	●	●
Suministrar nombres de archivos comodín como entrada de la asignación 402 1		●	●
Generar los nombres de los archivos de salida dinámicamente desde la asignación 402		●	●
Obtener valores de cadena de una asignación 156	●	●	●
Variables 160		●	●
Ordenar componentes 172		●	●
Funciones para agrupar datos 279		●	●
Filtros 178	●	●	●

Función	XSLT 1.0	XSLT 2.0	XSLT 3.0
Componentes Asignación de valores 184	●	●	●
Nombres de nodo dinámicos 385		●	●

Notas al pie:

1. XSLT 2.0, XSLT 3.0 y XQuery utilizan la función **`fn:collection`**. La implementación en los motores XSLT 2.0, XSLT 3.0 y XQuery de Altova resuelve comodines. Los demás motores pueden comportarse de modo distinto.

12.2 Información sobre motores

Esta sección ofrece información sobre las características de implementación del validador XML de Altova y de los motores XSLT 1.0, XSLT 2.0 y XQuery de Altova.

12.2.1 Información sobre motores XSLT y XQuery

Los motores XSLT y XQuery de MapForce 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. Por consiguiente, MapForce 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:

12.2.1.1 XSLT 1.0

El motor XSLT 1.0 de MapForce 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. Tenga en cuenta la información sobre la implementación que se ve a continuación.

Nota sobre la implementación

Cuando el atributo `method` de `xsl:output` tiene el valor HTML o si selecciona de forma predeterminada el formato de salida HTML, los caracteres especiales del archivo XML o XSLT se insertan en el documento HTML como referencias de caracteres HTML. Por ejemplo, el carácter U+00A0 (la referencia de carácter hexadecimal para un espacio de no separación) se inserta en el código HTML como referencia de carácter (` ` o ` `) o como referencia de entidad (` `).

12.2.1.2 XSLT 2.0

Temas de este apartado:

Especificaciones

El motor XSLT 2.0 de MapForce cumple con la recomendación XSLT 2.0 del 23 de enero de 2007 y la recomendación XPath 2.0 del 14 de diciembre de 2010, ambas del W3C.

Compatibilidad con versiones antiguas

El motor XSLT 2.0 es compatible con versiones previas. Esto es relevante cuando se utiliza el motor XSLT 2.0 para procesar una hoja de estilos o instrucción XSLT 1.0. Tenga en cuenta que los resultados obtenidos con el motor XSLT 1.0 pueden ser diferentes a los obtenidos con el motor XSLT 2.0 en modo de compatibilidad con versiones antiguas.

Espacios de nombres

En su hoja de estilos XSLT 2.0 debe declarar estos espacios de nombres para poder usar los constructores de tipo y las funciones disponibles en XSLT 2.0. Los prefijos que aparecen a continuación son los que se suelen usar, pero puede usar otros prefijos si quiere.

Espacio de nombres	Prefijo	URI del espacio de nombres
Tipos XML Schema	`xs:`	`http://www.w3.org/2001/XMLSchema`
Funciones XPath 2.0	`fn:`	`http://www.w3.org/2005/xpath-functions`

Estos espacios de nombres se suelen declarar en el elemento `xsl:stylesheet` o en el elemento `xsl:transform`:

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

Es necesario tener en cuenta que:

- El motor XSLT 2.0 utiliza el espacio de nombres Funciones XPath 2.0 y XQuery 1.0 como **espacio de nombres de funciones predeterminado**. Esto significa que puede usar funciones XPath 2.0 y XSLT 2.0 en su hoja de estilos sin prefijos. Si declara el espacio de nombres Funciones XPath 2.0 en su hoja de estilos con un prefijo, podrá usar el prefijo asignado en la declaración.

- Cuando se usan constructores de tipo y tipos del espacio de nombres XML Schema, el prefijo utilizado en la declaración de espacio de nombres se debe usar en la llamada al constructor de tipo (p.ej. `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 de algunas de las funciones XSLT 2.0 relacionadas con esta implementación.

`xsl:result-document`
También son compatibles estas codificaciones específicas de Altova: `x-base16tobinary` y `x-base64tobinary`.

`function-available`
Esta función mira si hay funciones del ámbito disponibles (funciones XSLT, XPath y de extensión).

`unparsed-text`
El atributo `href` acepta (i) rutas de acceso relativas para archivos que estén en la carpeta del URI base y (ii) rutas de acceso absolutas con o sin el protocolo `file://`. También son compatibles estas codificaciones específicas de Altova: `x-binarytobase16` y `x-binarytobase64`. Ejemplo: `xs:base64Binary(unparsed-text('chart.png', 'x-binarytobase64'))`.

`unparsed-text-available`
El argumento `href` acepta (i) rutas de acceso relativas para archivos que estén en la carpeta del URI base y (ii) rutas de acceso absolutas con o sin el protocolo `file://`. También son compatibles estas codificaciones específicas de Altova: `x-binarytobase16` y `x-binarytobase64`.

Nota: Estos valores de codificación estaban implementados en el ya descatalogado AltovaXML pero ya no se utilizan (son obsoletos): `base16tobinary`, `base64tobinary`, `binarytobase16` y `binarytobase64`.

12.2.1.3 XQuery 1.0

Temas de este apartado:

Especificaciones compatibles

El motor XQuery 1.0 de MapForce 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 MapForce.

Compatibilidad con esquemas

El motor XQuery 1.0 es **compatible con 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`
Instancia de esquema	`xsi:`	`http://www.w3.org/2001/XMLSchema-instance`
Funciones integradas	`fn:`	`http://www.w3.org/2005/xpath-functions`
Funciones locales	`local:`	`http://www.w3.org/2005/xquery-local-functions`

Es importante tener en cuenta que:

- El motor XQuery 1.0 entiende que los prefijos de la tabla anterior están enlazados con los correspondientes espacios de nombres.
- Como el espacio de nombres de funciones integradas (véase `fn:`) es el espacio de nombres de funciones predeterminado de XQuery, no es necesario usar el prefijo `fn:` cuando se invocan funciones integradas (p.ej. `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 (p.ej.: `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. (p.ej.: `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. (p.ej.: `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 tanto `xs:string` como `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[495]. Para usar una intercalación concreta, indique su URI tal y como aparece en la lista de intercalaciones compatibles[495]. Las comparaciones de cadenas de texto, incluidas las comparaciones para las funciones `fn:max` y `fn:min`, se harán según la intercalación especificada. Si no se indica la opción de intercalación, se utiliza la intercalación de puntos de código Unicode predeterminada.

Precisión de tipos numéricos

- El tipo de datos `xs:integer` es de precisión arbitraria, es decir, puede representar un número de dígitos cualquiera.
- El tipo de datos `xs:decimal` tiene un límite de 20 dígitos después del punto decimal.
- Los tipos de datos `xs:float` y `xs:double` tienen una precisión limitada de 15 dígitos.

Compatibilidad con instrucciones XQuery

La instrucción `Pragma` no es compatible. Si se encuentra, se ignora y en su lugar se evalúa la expresión de reserva.

Comportamiento propio de esta implementación

A continuación puede ver una descripción de cómo enfocan los motores XQuery y XQuery Update 1.0 los aspectos relativos a la implementación de ciertas funciones.

`unparsed-text`

El atributo `href` acepta (i) rutas de acceso relativas para archivos que estén en la carpeta del URI base y (ii) rutas de acceso absolutas con o sin el protocolo `file://`. También son compatibles estas codificaciones específicas de Altova: `x-binarytobase16` y `x-binarytobase64`. Ejemplo: `xs:base64Binary(unparsed-text('chart.png', 'x-binarytobase64'))`.

`unparsed-text-available`

El argumento `href` acepta (i) rutas de acceso relativas para archivos que estén en la carpeta del URI base y (ii) rutas de acceso absolutas con o sin el protocolo `file://`. También son compatibles estas codificaciones específicas de Altova: `x-binarytobase16` y `x-binarytobase64`.

Nota: Estos valores de codificación estaban implementados en el ya descatalogado AltovaXML pero ya no se utilizan (son obsoletos): `base16tobinary`, `base64tobinary`, `binarytobase16` y `binarytobase64`.

12.2.2 Funciones XSTL y XPath/XQuery

Esta sección enumera las funciones de extensión de Altova y otras funciones de extensión que se pueden utilizar con expresiones XPath y XQuery. Las funciones de extensión de Altova se pueden usar con los motores XSLT y XQuery de Altova y ofrecen algunas funciones más aparte de las que están disponibles en las bibliotecas de funciones definidas en los estándares del W3C.

En esta sección describimos principalmente las funciones de extensión XPath/XQuery que han sido creadas por Altova para proporcionar operaciones adicionales. Estas funciones[496] pueden ser calculadas por los motores XSLT y XQuery de Altova basándose en las reglas descritas en esta sección. Para obtener información sobre las funciones XPath/XQuery regulares, consulte la Referencia de funciones XPath/XQuery de Altova.

Aspectos generales

Es necesario tener en cuenta estos puntos generales:

- A las funciones de las bibliotecas de funciones principales definidas en las especificaciones W3C se les puede llamar sin un prefijo. Esto se debe a que los motores XSLT y XQuery leen funciones sin prefijo como si pertenecieran a un espacio de nombres de funciones predeterminado `http://www.w3.org/2005/xpath-functions`, que es el que se especifica en las especificaciones de las funciones XPath y XQuery. Si este espacio de nombres se declara explícitamente en un documento XSLT o XQuery, el prefijo utilizado en la declaración de espacio de nombres también se puede usar en el nombre de las funciones.
- Por lo general, si una función espera como argumento una secuencia de un elemento y se suministra una secuencia de más de un elemento, entonces se devuelve un error.
- Se usa la colación de punto de código de Unicode para todas las comparaciones de cadenas de texto.
- Los resultados que son QName se serializan de esta forma `[prefijo:]nombrelocal`.

Precisión de xs:decimal
La precisión se refiere a la cantidad de dígitos del número; la especificación requiere un mínimo de 18 dígitos. Para operaciones de división que dan un resultado de tipo `xs:decimal`, la precisión es de 19 dígitos tras el punto decimal sin redondeos.

Zona horaria implícita
Cuando hay que comparar dos valores `date`, `time` o `dateTime`, es necesario conocer el uso horario de los valores que se deben comparar. Cuando el uso horario no se conoce de forma explícita, se usa el uso horario implícito. La zona horaria implícita se toma del reloj del sistema y para probar cuál es su valor puede utilizar la función `implicit-timezone()`.

Intercalaciones
La colación predeterminada es la colación de punto de código de Unicode, que compara cadenas de texto basándose en su punto de código. El motor usa el algoritmo de colación de Unicode. Otras intercalaciones compatibles son las intercalaciones ICU que aparecen más abajo. Para usar una intercalación indique su URI tal y como aparece en la tabla más abajo. Las comparaciones de cadenas de texto (incluidas las que usan las funciones `max` y `min`) se harán según la intercalación especificada. Si no se ha indicado ninguna colación se usará la colación predeterminada de punto de código de Unicode.

Lenguaje	URIs
da: Danés	da_DK
de: Alemán	de_AT, de_BE, de_CH, de_DE, de_LI, de_LU
en: Inglés	en_AS, en_AU, en_BB, en_BE, en_BM, en_BW, en_BZ, en_CA, en_GB, en_GU, en_HK, en_IE, en_IN, en_JM, en_MH, en_MP, en_MT, en_MU, en_NA, en_NZ, en_PH, en_PK, en_SG, en_TT, en_UM, en_US, en_VI, en_ZA, en_ZW
es: Español	es_419, es_AR, es_BO, es_CL, es_CO, es_CR, es_DO, es_EC, es_ES, es_GQ, es_GT, es_HN, es_MX, es_NI, es_PA, es_PE, es_PR, es_PY, es_SV, es_US, es_UY, es_VE
fr: Francés	fr_BE, fr_BF, fr_BI, fr_BJ, fr_BL, fr_CA, fr_CD, fr_CF, fr_CG, fr_CH, fr_CI, fr_CM, fr_DJ, fr_FR, fr_GA, fr_GN, fr_GP, fr_GQ, fr_KM, fr_LU, fr_MC, fr_MF, fr_MG, fr_ML, fr_MQ, fr_NE, fr_RE, fr_RW, fr_SN, fr_TD, fr_TG
it: Italiano	it_CH, it_IT
ja: Japonés	ja_JP
nb: Noruego bokmål	nb_NO
nl: Holandés	nl_AW, nl_BE, nl_NL
nn: Noruego nynorsk	nn_NO
pt: Portugués	pt_AO, pt_BR, pt_GW, pt_MZ, pt_PT, pt_ST
ru: Ruso	ru_MD, ru_RU, ru_UA
sv: Sueco	sv_FI, sv_SE

Eje del espacio de nombres
El eje del espacio de nombres está obsoleto en XPath 2.0. Sin embargo, sí que se admite el uso del espacio de nombres. Para acceder a la información sobre el espacio de nombres con mecanismos de XPath 2.0, utilice las funciones `in-scope-prefixes()`, `namespace-uri()` y `namespace-uri-for-prefix()`.

12.2.2.1 Funciones de extensión de Altova

Puede utilizar todas las funciones de extensión de Altova en sus expresiones XPath/XQuery. Con ellas conseguirá funciones adicionales no disponibles en la biblioteca de funciones estándar de XPath, XQuery y XSLT. Las funciones de extensión de Altova están en el **espacio de nombres** **`http://www.altova.com/xslt-extensions`** y en esta sección se presentan con el prefijo, que se supone estará enlazado al espacio de nombres señalado. Tenga en cuenta que en futuras versiones del producto algunas funciones pueden dejar de ser compatibles o su comportamiento puede cambiar. Por tanto, consulte siempre la documentación del producto para conocer el funcionamiento de estas funciones en cada versión del producto.

Las funciones definidas en las especificaciones XPath/XQuery Functions del W3C se pueden usar en (i) expresiones XPath en contextos XSLT y en (ii) expresiones XQuery en documentos XQuery. En esta documentación las funciones que se pueden usar en el primer contexto (XPath en XSLT) llevan el símbolo **XP** y se les llama funciones XPath. Las funciones que se pueden usar en contextos XQuery llevan el símbolo **XQ** y funcionan como funciones XQuery. Las especificaciones XSLT del W3C también definen funciones que se pueden usar en expresiones XPath en documentos XSLT. Estas funciones llevan el símbolo **XSLT** y se les denomina funciones XSLT. Por cada función se indica en qué versión de XPath/XQuery y XSLT se puede usar (ver símbolos más abajo). Las funciones de las bibliotecas de funciones XPath/XQuery y XSLT aparecen sin prefijo. Las funciones de extensión de otras bibliotecas, como las funciones de extensión de Altova, aparecen con un prefijo.

Funciones XPath *(en expresiones XPath en XSLT):*	**XP1** **XP2** **XP3.1.1**
Funciones XSLT *(en expresiones XPath en XSLT):*	**XSLT1** **XSLT2** **XSLT3**
Funciones XQuery *(en expresiones XQuery en XQuery):*	**XQ1** **XQ3.1**

Cómo usar las funciones de extensión de Altova

Para poder usar las funciones de extensión de Altova debe declarar el espacio de nombre correspondiente (*el primer resaltado en el extracto de código siguiente*) y después usar las funciones de extensión para que se resuelvan como si pertenecieran a ese espacio de nombres (*véase el segundo resaltado*). En el ejemplo siguiente puede ver cómo se usa la función de extensión de Altova **age**.

```
<xsl:stylesheet version="2.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:fn="http://www.w3.org/2005/xpath-functions"
    xmlns:altova="http://www.altova.com/xslt-extensions">
    <xsl:output method="text" encoding="ISO-8859-1"/>
    <xsl:template match="Persons">
        <xsl:for-each select="Person">
            <xsl:value-of select="concat(Name, ': ')"/>
            <xsl:value-of select="altova:age(xs:date(BirthDate))"/>
            <xsl:value-of select="' years&#x0A;'"/>
        </xsl:for-each>
    </xsl:template>
</xsl:stylesheet>
```

Funciones XSLT 498

Las funciones XSLT solo se pueden utilizar en expresiones XPath en un contexto XSLT (igual que las funciones XSLT 2.0 `current-group()` o `key()`). Estas funciones no están pensadas para contextos no XSLT (p. ej. contextos XQuery) y, por tanto, no funcionarán en contextos que no sean XSLT. Recuerde que las funciones XSLT para XBRL solamente se pueden utilizar con ediciones de los productos de Altova compatibles con XBRL.

Funciones XPath/XQuery

Las funciones XPath/XQuery se pueden utilizar en expresiones XPath, en contextos XSLT y en expresiones XQuery:

12.2.2.1.1 Funciones XSLT

Las **funciones de extensión XSLT** pueden utilizarse en expresiones XPath en contextos XSLT y no funcionan en contextos que no sean XSLT (por ejemplo, en contextos XQuery).

Nota sobre el nombre de las funciones y lenguajes

Puede utilizar todas las funciones de extensión de Altova en sus expresiones XPath/XQuery. Con ellas conseguirá funciones adicionales no disponibles en la biblioteca de funciones estándar de XPath, XQuery y XSLT. Las funciones de extensión de Altova están en el **espacio de nombres** **`http://www.altova.com/xslt-extensions`** y en esta sección se presentan con el prefijo, que se supone estará enlazado al espacio de nombres señalado. Tenga en cuenta que en futuras versiones del producto algunas funciones pueden dejar de ser compatibles o su comportamiento puede cambiar. Por tanto, consulte siempre la documentación del producto para conocer el funcionamiento de estas funciones en cada versión del producto.

Funciones XPath (en expresiones XPath en XSLT):	XP1 XP2 XP3.1.1
Funciones XSLT (en expresiones XPath en XSLT):	XSLT1 XSLT2 XSLT3
Funciones XQuery (en expresiones XQuery en XQuery):	XQ1 XQ3.1

Funciones generales

▼ distinct-nodes [altova:]

`altova:distinct-nodes(`*`node()*`***`)`** como **`node()*`** XSLT1 XSLT2 XSLT3

Toma un conjunto de nodos como entrada y devuelve el mismo conjunto menos los nodos que tengan el mismo valor (es decir, devuelve los nodos que son únicos). La comparación se hace con la función XPath/XQuery `fn:deep-equal`.

− *Ejemplo*

- **`altova:distinct-nodes`**`(country)` devuelve todos los nodos secundarios `country` excepto los

que tengan el mismo valor.

▼ evaluate [altova:]

altova:evaluate(ExpresiónXPath *como xs:string***[, ValorDe$p1, ... ValorDe$pN])** XSLT1 XSLT2 XSLT3

Toma una expresión XPath, pasada como cadena, como argumento obligatorio. Devuelve el resultado de la expresión evaluada. Por ejemplo, **evaluate('//Name[1]')** devuelve el contenido del primer elemento `Name` del documento. Observe que para pasar la expresión `//Name[1]` como cadena basta con ponerla entre comillas simples.

La función `altova:evaluate` puede tomar más argumentos, que son los valores de las variables del ámbito que se llaman `p1`, `p2`, `p3`... `pN`. Recuerde que (i) las variables deben definirse con nombres de tipo `pX`, siendo `X` un entero; (ii) los argumentos de la función `altova:evaluate` (*ver firma más abajo*), a partir del segundo argumento, ofrecen los valores de las variables, correspondiendo la secuencia de argumentos a la secuencia numérica de variables: `p1` corresponde a `pN` y el segundo argumento será el valor de la variable `p1`, el tercer argumento al de la variable `p2`, y así sucesivamente; (iii) los valores de las variables deben ser de tipo `item*`.

Ejemplo

```
<xsl:variable name="xpath" select="'$p3, $p2, $p1'" />
<xsl:value-of select="altova:evaluate($xpath, 10, 20, 'hi')" />
```

da el resultado `"hi 20 10"`

En el ejemplo anterior puede observar que:

- El segundo argumento de la expresión `altova:evaluate` es el valor asignado a la variable `$p1`, el tercer argumento es el valor asignado a la variable `$p2` y así sucesivamente.
- Observe que el cuarto argumento de la función es un valor de cadena porque va entre comillas simples.
- El atributo `select` del elemento `xs:variable` suministra la expresión XPath. Como esta expresión debe ser de tipo `xs:string`, se pone entre comillas simples.

Más ejemplos

- ```
  <xsl:variable name="xpath" select="'$p1'" />
  <xsl:value-of select="altova:evaluate($xpath, //Name[1])" />
  ```
  El resultado es el valor del primer elemento `Name`.

- ```
  <xsl:variable name="xpath" select="'$p1'" />
  <xsl:value-of select="altova:evaluate($xpath, '//Name[1]')" />
  ```
  El resultado es "`//Name[1]`*"*

La función de extensión `altova:evaluate()` es muy práctica cuando una expresión XPath de la hoja de estilos XSLT contiene partes que se deben evaluar de forma dinámica. Por ejemplo, imagine que el usuario selecciona un criterio de ordenación y este criterio se almacena en el atributo `UserReq/@sortkey`. En la hoja de estilos podría tener esta expresión:

```
<xsl:sort select="altova:evaluate(../UserReq/@sortkey)" order="ascending"/>
```

La función `altova:evaluate()` lee el atributo `sortkey` del elemento secundario `UserReq` del primario del nodo de contexto. Imagine que el valor del atributo `sortkey` es `Price`. En ese caso, la función `altova:evaluate()` devuelve `Price`, que se convierte en el valor del atributo `select`:

```
<xsl:sort select="Price" order="ascending"/>
```

Si esta instrucción `sort` aparece dentro del contexto de un elemento llamado `Order`, entonces los elementos `Order` se ordenan según el valor de los secundarios `Price`. Otra opción es que, si el valor de `@sortkey` fuera `Date`, por ejemplo, entonces los elementos `Order` se ordenarían según el valor de los secundarios `Date`. Es decir, el criterio de ordenación para `Order` se selecciona del atributo `sortkey` en tiempo de ejecución. Esto no sería posible con una expresión como:

```
<xsl:sort select="../UserReq/@sortkey" order="ascending"/>
```

En este caso, el criterio de ordenación sería el propio atributo `sortkey`, no `Price` ni `Date` (ni otro contenido actual de `sortkey`).

Nota: el contexto estático incluye espacios de nombres, tipos y funciones (pero no variables) del entorno de llamada. El URI base y el espacio de nombres predeterminado se heredan.

Más ejemplos

- Variables estáticas: `<xsl:value-of select="$i3, $i2, $i1" />`
  El resultado es los valores de las tres variables.

- Expresión XPath dinámica con variables dinámicas:
  `<xsl:variable name="xpath" select="'$p3, $p2, $p1'" />`
  `<xsl:value-of select="altova:evaluate($xpath, 10, 20, 30)" />`
  El resultado es "30 20 10"

- Expresión XPath dinámica sin variables dinámicas:
  `<xsl:variable name="xpath" select="'$p3, $p2, $p1'" />`
  `<xsl:value-of select="altova:evaluate($xpath)" />`
  Error: no se definió la variable para $p3.

encode-for-rtf [altova:]

altova:encode-for-rtf(entrada *como xs:string*, **conservarEspaciosEnBlanco** *como xs:boolean*, **conservarLíneasNuevas** *como xs:boolean***)** como **xs:string** XSLT2 XSLT3
Convierte la cadena de entrada en código para RTF. Los espacios en blanco y las líneas nuevas se conservan o no dependiendo del valor booleano especificado para los correspondientes parámetros.

[Subir 498]

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 498]

12.2.2.1.2 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 y están disponibles en contextos XPath/XQuery.

Nota sobre el nombre de las funciones y lenguajes

Puede utilizar todas las funciones de extensión de Altova en sus expresiones XPath/XQuery. Con ellas conseguirá funciones adicionales no disponibles en la biblioteca de funciones estándar de XPath, XQuery y XSLT. Las funciones de extensión de Altova están en el **espacio de nombres** **http://www.altova.com/xslt-extensions** y en esta sección se presentan con el prefijo, que se supone estará enlazado al espacio de nombres señalado. Tenga en cuenta que en futuras versiones del producto algunas funciones pueden dejar de ser compatibles o su comportamiento puede cambiar. Por tanto, consulte siempre la documentación del producto para conocer el funcionamiento de estas funciones en cada versión del producto.

Funciones XPath (en expresiones XPath en XSLT):	XP1 XP2 XP3.1.1
Funciones XSLT (en expresiones XPath en XSLT):	XSLT1 XSLT2 XSLT3
Funciones XQuery (en expresiones XQuery en XQuery):	XQ1 XQ3.1

▼ Funciones agrupadas según su funcionalidad

- Devolver el número de semana de una fecha como número entero 511
- Generar la fecha, la hora y el tipo de duración a partir de los componentes léxicos de cada tipo 512
- Construir un tipo date, dateTime o a partir de la cadena de entrada 514
- Funciones para calcular la edad 516
- Funciones para calcular el tiempo Unix 517

▼ Funciones por orden alfabético

[Subir 501]

Agregar una duración a xs:dateTime XP3.1 XQ3.1

Estas funciones sirven para agregar una duración a **xs:dateTime** y devuelven **xs:dateTime**. El tipo xs:dateTime tiene el formato SSAA-MM-DDThh:mm:ss.sss. Se trata de la concatenación de los formatos

`xs:date` y `xs:time` separados por la letra `T`. Si quiere puede usar un sufijo de zona horaria (por ejemplo **`+01:00`**).

▼ add-years-to-dateTime [altova:]

`altova:add-years-to-dateTime(FechaHora` *`as xs:dateTime`*`,` **`Años`** *`as xs:integer`***`)`** como **`xs:dateTime`** **XP3.1** **XQ3.1**
Añade una duración en años un valor de fecha y hora. El segundo argumento es el número de años que se debe añadir al valor de fecha y hora dado como primer argumento. El resultado es de tipo `xs:dateTime`.

⊟ *Ejemplos*

- **`altova:add-years-to-dateTime`**`(xs:dateTime("2014-01-15T14:00:00"), 10)` devuelve `2024-01-15T14:00:00`
- **`altova:add-years-to-dateTime`**`(xs:dateTime("2014-01-15T14:00:00"), -4)` devuelve `2010-01-15T14:00:00`

▼ add-months-to-dateTime [altova:]

`altova:add-months-to-dateTime(FechaHora` *`as xs:dateTime`*`,` **`Meses`** *`as xs:integer`***`)`** como **`xs:dateTime`** **XP3.1** **XQ3.1**
Añade una duración en meses a un valor de fecha y hora. El segundo argumento es el número de meses que se debe añadir al valor de fecha y hora dado como primer argumento. El resultado es de tipo `xs:dateTime`.

⊟ *Ejemplos*

- **`altova:add-months-to-dateTime`**`(xs:dateTime("2014-01-15T14:00:00"), 10)` devuelve `2014-11-15T14:00:00`
- **`altova:add-months-to-dateTime`**`(xs:dateTime("2014-01-15T14:00:00"), -2)` devuelve `2013-11-15T14:00:00`

▼ add-days-to-dateTime [altova:]

`altova:add-days-to-dateTime(FechaHora` *`as xs:dateTime`*`,` **`Días`** *`as xs:integer`***`)`** como **`xs:dateTime`** **XP3.1** **XQ3.1**
Añade una duración en días a un valor de fecha y hora. El segundo argumento es el número de días que se deben añadir al valor de fecha y hora dado como primer argumento. El resultado es de tipo `xs:dateTime`.

⊟ *Ejemplos*

- **`altova:add-days-to-dateTime`**`(xs:dateTime("2014-01-15T14:00:00"), 10)` devuelve `2014-01-25T14:00:00`
- **`altova:add-days-to-dateTime`**`(xs:dateTime("2014-01-15T14:00:00"), -8)` devuelve `2014-01-07T14:00:00`

▼ add-hours-to-dateTime [altova:]

`altova:add-hours-to-dateTime(FechaHora` *`as xs:dateTime`*`,` **`Horas`** *`as xs:integer`***`)`** como

xs:dateTime XP3.1 XQ3.1
Añade una duración en horas a un valor de fecha y hora. El segundo argumento es el número de horas que se deben añadir al valor de fecha y hora dado como primer argumento. El resultado es de tipo `xs:dateTime`.

Ejemplos

- **altova:add-hours-to-dateTime**(xs:dateTime("2014-01-15T13:00:00"), 10) devuelve 2014-01-15T23:00:00
- **altova:add-hours-to-dateTime**(xs:dateTime("2014-01-15T13:00:00"), -8) devuelve 2014-01-15T05:00:00

add-minutes-to-dateTime [altova:]

altova:add-minutes-to-dateTime(FechaHora *as xs:dateTime*, **Minutos** *as xs:integer***)** como **xs:dateTime** XP3.1 XQ3.1
Añade una duración en minutos a un valor de fecha y hora. El segundo argumento es el número de minutos que se debe añadir al valor de fecha y hora dado como primer argumento. El resultado es de tipo `xs:dateTime`.

Ejemplos

- **altova:add-minutes-to-dateTime**(xs:dateTime("2014-01-15T14:10:00"), 45) devuelve 2014-01-15T14:55:00
- **altova:add-minutes-to-dateTime**(xs:dateTime("2014-01-15T14:10:00"), -5) devuelve 2014-01-15T14:05:00

add-seconds-to-dateTime [altova:]

altova:add-seconds-to-dateTime(FechaHora *as xs:dateTime*, **Segundos** *as xs:integer***)** como **xs:dateTime** XP3.1 XQ3.1
Añade una duración en segundos a un valor de fecha y hora. El segundo argumento es el número de segundos que se debe añadir al valor de fecha y hora dado como primer argumento. El resultado es de tipo `xs:dateTime`.

Ejemplos

- **altova:add-seconds-to-dateTime**(xs:dateTime("2014-01-15T14:00:10"), 20) devuelve 2014-01-15T14:00:30
- **altova:add-seconds-to-dateTime**(xs:dateTime("2014-01-15T14:00:10"), -5) devuelve 2014-01-15T14:00:05

[Subir 501]

Agregar una duración a xs:date XP3.1 XQ3.1

Estas funciones agregan una duración a **xs:date** y devuelven **xs:date**. El tipo `xs:date` tiene el formato `SSAA-MM-DD`.

add-years-to-date [altova:]

altova:add-years-to-date(Fecha *as xs:date*, **Años** *as xs:integer*) como **xs:date** XP3.1 XQ3.1
Añade una duración en años a una fecha. El segundo parámetro es el número de años que se debe añadir a la fecha dada como primer argumento. El resultado es de tipo `xs:date`.

Ejemplos

- **altova:add-years-to-date**(xs:date("2014-01-15"), 10) devuelve 2024-01-15
- **altova:add-years-to-date**(xs:date("2014-01-15"), -4) devuelve 2010-01-15

add-months-to-date [altova:]

altova:add-months-to-date(Fecha *as xs:date*, **Meses** *as xs:integer*) como **xs:date** XP3.1 XQ3.1
Añade una duración en meses a una fecha. El segundo argumento es el número de meses que se debe añadir a la fecha dada como primer argumento. El resultado es de tipo `xs:date`.

Ejemplos

- **altova:add-months-to-date**(xs:date("2014-01-15"), 10) devuelve 2014-11-15
- **altova:add-months-to-date**(xs:date("2014-01-15"), -2) devuelve 2013-11-15

add-days-to-date [altova:]

altova:add-days-to-date(Fecha *as xs:date*, **Días** *as xs:integer*) como **xs:date** XP3.1 XQ3.1
Añade una duración en días a una fecha. El segundo argumento es el número de días que se deben añadir a la fecha dad como primer argumento. El resultado es de tipo `xs:date`.

Ejemplos

- **altova:add-days-to-date**(xs:date("2014-01-15"), 10) devuelve 2014-01-25
- **altova:add-days-to-date**(xs:date("2014-01-15"), -8) devuelve 2014-01-07

[Subir 501]

Recuperar duraciones y aplicarles formato XP3.1 XQ3.1

Estas funciones analizan la entrada **xs:duration** o **xs:string** y devuelven, respectivamente, **xs:string** o **xs:duration**.

format-duration [altova:]

altova:format-duration(Duración *como xs:duration*, **Imagen** *como xs:string*) como **xs:string** XP3.1 XQ3.1
Aplica formato a una duración, que se suministra como primer argumento, en base a la cadena de imagen dada como segundo argumento. El resultado es una cadena de texto cuyo formato se ajusta a la cadena de imagen.

Ejemplos

- **altova:format-duration**(xs:duration("P2DT2H53M11.7S"), "Días:[D01] Horas:[H01] Minutos:[m01] Segundos:[s01] Fracciones:[f0]") devuelve "Días:02 Horas:02 Minutos:53 Segundos:11 Fracciones:7"

- **altova:format-duration**(xs:duration("P3M2DT2H53M11.7S"), "Meses:[M01] Días:[D01] Horas:[H01] Minutos:[m01]") devuelve "Meses:03 Días:02 Horas:02 Minutos:53"

parse-duration [altova:]

altova:parse-duration(CadenaEntrada *como xs:string*, **Imagen** *como xs:string*) como **xs:duration** XP3.1 XQ3.1
Toma una cadena con patrón como primer argumento y una cadena de imagen como segundo argumento. La cadena de entrada se analiza en base a la cadena de imagen y se devuelve un xs:duration.

Ejemplos

- **altova:parse-duration**("Días:02 Horas:02 Minutos:53 Segundos:11 Fracciones:7"), "Días:[D01] Horas:[H01] Minutes:[m01] Segundos:[s01] Fracciones:[f0]") devuelve "P2DT2H53M11.7S"
- **altova:parse-duration**("Meses:03 Días:02 Horas:02 Minutos:53 Segundos:11 Fracciones:7", "Meses:[M01] Días:[D01] Horas:[H01] Minutos:[m01]") devuelve "P3M2DT2H53M"

[Subir 501]

Agregar una duración a xs:time XP3.1 XQ3.1

Estas funciones agregan una duración a **xs:time** y devuelven **xs:time**. El tipo xs:time tiene un formato léxico de este tipo hh:mm:ss.sss. Si quiere, puede añadir un sufijo de zona horaria. La letra Z indica (UTC). Las demás zonas horarias se representan con la diferencia que hay entre ellas y la zona UTC: +hh:mm o -hh:mm. Si falta el valor de zona horaria, se entiende que se desconoce (no se da por hecho que es UTC)

add-hours-to-time [altova:]

altova:add-hours-to-time(Hora *as xs:time*, **Horas** *as xs:integer*) como **xs:time** XP3.1 XQ3.1
Añade una duración en horas a una hora. El segundo argumento es el número de horas que se debe añadir a la hora dada como primer argumento. El resultado es de tipo xs:time.

Ejemplos

- **altova:add-hours-to-time**(xs:time("11:00:00"), 10) devuelve 21:00:00
- **altova:add-hours-to-time**(xs:time("11:00:00"), -7) devuelve 04:00:00

add-minutes-to-time [altova:]

altova:add-minutes-to-time(Hora *as xs:time*, **Minutos** *as xs:integer*) como **xs:time** XP3.1 XQ3.1
Añade una duración en minutos a una hora. El segundo argumento es el número de minutos que se debe añadir a la hora dada como primer argumento. El resultado es de tipo xs:time.

Ejemplos

- **altova:add-minutes-to-time**(xs:time("14:10:00"), 45) devuelve 14:55:00
- **altova:add-minutes-to-time**(xs:time("14:10:00"), -5) devuelve 14:05:00

▼ add-seconds-to-time [altova:]

altova:add-seconds-to-time(Hora *as xs:time*, **Segundos** *as xs:integer*) como **xs:time** XP3.1 XQ3.1
Añade una duración en segundos a una hora. El segundo argumento es el número de segundos que se debe añadir a la hora dada como primer argumento. El resultado es de tipo `xs:time`. El componente **Segundos** puede estar comprendido entre `0` y `59.999`.

▣ *Ejemplos*

- **altova:add-seconds-to-time**(xs:time("14:00:00"), 20) devuelve 14:00:20
- **altova:add-seconds-to-time**(xs:time("14:00:00"), 20.895) devuelve 14:00:20.895

[Subir 501]

Quitar la parte de zona horaria de los tipos de datos date/time XP3.1 XQ3.1

Estas funciones quitan la zona horaria de los valores **xs:dateTime**, **xs:date** o **xs:time** actuales. Tenga en cuenta que la diferencia entre `xs:dateTime` y `xs:dateTimeStamp` es que en esta última la parte de zona horaria es obligatoria (mientras que en la primera es opcional). Es decir, el formato de un valor `xs:dateTimeStamp` puede ser `SSAA-MM-DDThh:mm:ss.sss±hh:mm` o `SSAA-MM-DDThh:mm:ss.sssZ`. Si la fecha y la hora se leen del reloj del sistema como `xs:dateTimeStamp`, la función `current-dateTime-no-TZ()` se puede usar para quitar la zona horaria.

▼ current-date-no-TZ [altova:]

altova:current-date-no-TZ() como **xs:date** XP3.1 XQ3.1
Esta función no toma ningún argumento. Quita la parte de zona horaria de la función `current-date()` (que es la fecha actual según el reloj del sistema) y devuelve un valor de tipo `xs:date`.

▣ *Ejemplos*

Si la fecha actual es **2014-01-15+01:00**:

- **altova:current-date-no-TZ**() devuelve 2014-01-15

▼ current-dateTime-no-TZ [altova:]

altova:current-dateTime-no-TZ() como **xs:dateTime** XP3.1 XQ3.1
Esta función no toma ningún argumento. Quita la parte de zona horaria de `current-dateTime()` (que es la fecha y hora actual según el reloj del sistema) y devuelve un valor de tipo `xs:dateTime`.

▣ *Ejemplos*

Si la fecha y hora actual es **2014-01-15T14:00:00+01:00**:

- **altova:current-dateTime-no-TZ**() devuelve 2014-01-15T14:00:00

▼ current-time-no-TZ [altova:]

altova:current-time-no-TZ() asxs:time XP3.1 XQ3.1
Esta función no toma ningún argumento. Quita la parte de zona horaria de `current-time()` (que es la hora actual según el reloj del sistema) y devuelve un valor de tipo `xs:time`.

Ejemplos

Si la hora actual es **14:00:00+01:00**:

- **altova:current-time-no-TZ**() devuelve `14:00:00`

date-no-TZ [altova:]

altova:date-no-TZ(InputDate *as xs:date***)** como **xs:date** XP3.1 XQ3.1
Esta función toma un argumento `xs:date`, del que elimina la parte `timezone` y devuelve un valor `xs:date`. Observe que la fecha permanece intacta.

Ejemplos

- **altova:date-no-TZ**(xs:date("2014-01-15+01:00")) devuelve `2014-01-15`

dateTime-no-TZ [altova:]

altova:dateTime-no-TZ(InputDateTime *as xs:dateTime***)** como **xs:dateTime** XP3.1 XQ3.1
Esta función toma un argumento `xs:dateTime`, del que elimina la parte `timezone`, y devuelve un valor `xs:dateTime`. Observe que tanto la fecha como la hora permanecen intactas.

Ejemplos

- **altova:dateTime-no-TZ**(xs:date("2014-01-15T14:00:00+01:00")) devuelve `2014-01-15T14:00:00`

time-no-TZ [altova:]

altova:time-no-TZ(HoraEntrada *como xs:time***)** como **xs:time** XP3.1 XQ3.1
Esta función toma un argumento `xs:time`, quita la parte de la zona horaria y devuelve un valor `xs:time`. Tenga en cuenta que la hora no se modifica.

Ejemplos

- **altova:time-no-TZ**(xs:time("14:00:00+01:00")) devuelve `14:00:00`

[Subir 501]

Devolver el número de días, horas, minutos y segundos de duraciones XP3.1 XQ3.1

Estas funciones devuelven el número de días en un mes y el número de horas, minutos y segundos de las duraciones correspondientes.

days-in-month [altova:]

altova:days-in-month(Year *as xs:integer*, **Month** *as xs:integer*) como **xs:integer** XP3.1 XQ3.1
Devuelve el número de días en el mes indicado. El mes se indica con los argumentos `Year` y `Month`.

Ejemplos

- **altova:days-in-month**(2018, 10) devuelve 31
- **altova:days-in-month**(2018, 2) devuelve 28
- **altova:days-in-month**(2020, 2) devuelve 29

hours-from-dayTimeDuration-accumulated

altova:hours-from-dayTimeDuration-accumulated(DayAndTime *como xs:duration*) como **xs:integer** XP3.1 XQ3.1
Devuelve el número total de horas de la duración enviada por el argumento `DayAndTime` (que es de tipo `xs:duration`). Las horas de los componentes `Day` y `Time` se agregan juntos para dar como resultado un número entero. Una hora nueva son 60 minutos enteros. Las duraciones negativas dan como resultado un valor de hora negativo.

Ejemplos

- **altova:hours-from-dayTimeDuration-accumulated**(xs:duration("P5D") devuelve 120, que es el número total de horas en 5 días.
- **altova:hours-from-dayTimeDuration-accumulated**(xs:duration("P5DT2H") devuelve 122, que es el número total de horas en 5 días más 2 horas.
- **altova:hours-from-dayTimeDuration-accumulated**(xs:duration("P5DT2H60M") devuelve 123, que es el número total de horas en 5 días más 2 horas y 60 mins.
- **altova:hours-from-dayTimeDuration-accumulated**(xs:duration("P5DT2H119M") devuelve 123, que es el número total de horas en 5 días más 2 horas y 119 mins.
- **altova:hours-from-dayTimeDuration-accumulated**(xs:duration("P5DT2H120M") devuelve 124, que es el número total de horas en 5 días más 2 horas y 120 mins.
- **altova:hours-from-dayTimeDuration-accumulated**(xs:duration("-P5DT2H") devuelve -122

minutes-from-dayTimeDuration-accumulated

altova:minutes-from-dayTimeDuration-accumulated(DayAndTime *como xs:duration*) como **xs:integer** XP3.1 XQ3.1
Devuelve el número total de minutos de la duración enviada por el argumento `DayAndTime` (que es de tipo `xs:duration`). Los minutos de los componentes `Day` y `Time` se agregan juntos para dar como resultado un número entero. Las duraciones negativas dan como resultado un valor de minutos negativo.

Ejemplos

- **altova:minutes-from-dayTimeDuration-accumulated**(xs:duration("PT60M") devuelve 60
- **altova:minutes-from-dayTimeDuration-accumulated**(xs:duration("PT1H") devuelve 60, que es el número total de minutos en 1 hora.
- **altova:minutes-from-dayTimeDuration-accumulated**(xs:duration("PT1H40M") devuelve 100
- **altova:minutes-from-dayTimeDuration-accumulated**(xs:duration("P1D") devuelve 1440, que es el número total de minutos en 1 día.
- **altova:minutes-from-dayTimeDuration-accumulated**(xs:duration("-P1DT60M") devuelve -1500

▼ seconds-from-dayTimeDuration-accumulated

altova:seconds-from-dayTimeDuration-accumulated(DayAndTime *como xs:duration*) como **xs:integer** XP3.1 XQ3.1
Devuelve el número total de segundos de la duración enviada por el argumento `DayAndTime` (que es de tipo `xs:duration`). Los segundos de los componentes Day y Time se agregan juntos para dar como resultado un número entero. Las duraciones negativas dan como resultado un valor de segundos negativo.

⊞ *Ejemplos*

- `altova:seconds-from-dayTimeDuration-accumulated(xs:duration("PT1M")` devuelve `60`, que es el número total de segundos en 1 minuto.
- `altova:seconds-from-dayTimeDuration-accumulated(xs:duration("PT1H")` devuelve `3600`, que es el número total de segundos en 1 hora.
- `altova:seconds-from-dayTimeDuration-accumulated(xs:duration("PT1H2M")` devuelve `3720`
- `altova:seconds-from-dayTimeDuration-accumulated(xs:duration("P1D")` devuelve `86400`, que es el número total de segundos en 1 día.
- `altova:seconds-from-dayTimeDuration-accumulated(xs:duration("-P1DT1M")` devuelve `-86460`

Obtener el día de la semana de `xs:dateTime` o `xs:date` XP3.1 XQ3.1

Estas funciones obtienen el día de la semana (como entero) de `xs:dateTime` o `xs:date`. Los días de la semana se numeran del `1` al `7` (usando el formato EE UU, es decir `Domingo =1`). En el formato europeo la semana empieza el lunes (es decir, `Lunes=1`). Para establecer el formato EE UU (`Domingo=1`) use el entero `0` allí donde se acepte un entero para indicar el formato.

▼ weekday-from-dateTime [altova:]

altova:weekday-from-dateTime(DateTime *como xs:dateTime*) **como xs:integer** XP3.1 XQ3.1
Toma una fecha como único argumento y devuelve el día de la semana de la fecha dada como número entero. Los días de la semana se numeran del 1 al 7 empezando por `Domingo=1`. Si necesita usar el formato europeo (donde `Lunes=1`), utilice la otra firma de esta función (*ver más abajo*).
⊟ *Ejemplos*

- `altova:weekday-from-dateTime(xs:dateTime("2014-02-03T09:00:00"))` devuelve `2`, lo cual significa "Lunes"..

altova:weekday-from-dateTime(DateTime *como xs:dateTime*, **Formato** *como xs:integer*) **como xs:integer** XP3.1 XQ3.1
Toma una fecha como primer argumento y devuelve el día de la semana de la fecha dada como número entero. Si el segundo argumento (número entero) es `0`, entonces los días de la semana se numeran del `1` al `7` empezando por `Domingo=1`. Si el segundo argumento es un entero distinto de `0`, entonces `Lunes=1`. Si falta el segundo argumento, la función se lee como en la firma anterior (*ver más arriba*).
⊟ *Ejemplos*

- `altova:weekday-from-dateTime(xs:dateTime("2014-02-03T09:00:00"), 1)` devuelve `1`, lo cual significa "Lunes"

- `altova:weekday-from-dateTime(xs:dateTime("2014-02-03T09:00:00"), 4)` devuelve `1`, lo cual significa "Lunes"
- `altova:weekday-from-dateTime(xs:dateTime("2014-02-03T09:00:00"), 0)` devuelve `2`, lo cual significa "Lunes"

▼ weekday-from-date [altova:]

`altova:weekday-from-date(Date` *como xs:date*`)` **como xs:integer** XP3.1 XQ3.1
Toma una fecha como único argumento y devuelve el día de la semana de la fecha dada como número entero. Los días de la semana se numeran del 1 al 7 empezando por `Domingo=1`. Si necesita usar el formato europeo (donde `Lunes=1`), utilice la otra firma de esta función (*ver más abajo*).

Ejemplos

- `altova:weekday-from-date(xs:date("2014-02-03+01:00"))` devuelve `2`, lo cual significa lunes.

`altova:weekday-from-date(Date` *como xs:date*`, Formato` *como xs:integer*`)` **como xs:integer** XP3.1 XQ3.1
Toma una fecha como primer argumento y devuelve el día de la semana de la fecha dada como número entero. Si el segundo argumento (`Formato`) es `0`, entonces los días de la semana se numeran del `1` al `7` empezando por `Domingo=1`. Si el segundo argumento es un entero distinto de `0`, entonces `Lunes=1`. Si falta el segundo argumento, la función se lee como en la firma anterior (*ver más arriba*).

Ejemplos

- `altova:weekday-from-date(xs:date("2014-02-03"), 1)` devuelve `1`, lo cual significa "Lunes"
- `altova:weekday-from-date(xs:date("2014-02-03"), 4)` devuelve `1`, lo cual significa "Lunes"
- `altova:weekday-from-date(xs:date("2014-02-03"), 0)` devuelve `2`, lo cual significa "Lunes".

[Subir 501]

Devolver el número de semana de xs:dateTime o xs:date XP2 XQ1 XP3.1 XQ3.1

Estas funciones devuelven el número de semana (como número entero) de `xs:dateTime` o `xs:date`. El número de la semana está disponible en el formato de calendario estadounidense, europeo e islámico. La razón de que los números de semana difieran en cada uno de estos calendarios es que en cada uno de ellos se considera un día diferente para el inicio de la semana (p. ej. en el formato estadounidense el primer día de la semana es el domingo).

▼ weeknumber-from-date [altova:]

`altova:weeknumber-from-date(Fecha` *como xs:date*`, Calendario` *como xs:integer*`)` como **xs:integer** XP2 XQ1 XP3.1 XQ3.1
Devuelve como número entero el número de semana del argumento **`Fecha`** dado. El segundo argumento (**`Calendario`**) indica el sistema de calendario que se debe seguir.
Estos son los valores permitidos para el argumento **`Calendario`**:

- **`0` = Calendario estadounidense** (la semana comienza el domingo)
- **`1` = Calendario estándar ISO o europeo** (la semana comienza el lunes)
- **`2` = Calendario islámico** (la semana comienza el sábado)

El valor predeterminado es **`0`**.

Ejemplos

- `altova:weeknumber-from-date(xs:date("2014-03-23"), 0)` devuelve `13`
- `altova:weeknumber-from-date(xs:date("2014-03-23"), 1)` devuelve `12`
- `altova:weeknumber-from-date(xs:date("2014-03-23"), 2)` devuelve `13`
- `altova:weeknumber-from-date(xs:date("2014-03-23") )` devuelve `13`

El día de la fecha de los ejemplos anteriores (`2014-03-23`) es un domingo. Por tanto, en este caso, el calendario estadounidense y el islámico van una semana por delante del calendario europeo.

weeknumber-from-dateTime [altova:]

altova:weeknumber-from-dateTime(FechaHora *como xs:dateTime*, **Calendario** *como xs:integer***)** como **xs:integer** XP2 XQ1 XP3.1 XQ3.1

Devuelve como entero el día de la semana del argumento **FechaHora** dado. El segundo argumento (**Calendario**) indica el sistema de calendario que se debe seguir.
Estos son los valores permitidos para el argumento **Calendario**:

- **0 = Calendario estadounidense** (la semana comienza el domingo)
- **1 = Calendario estándar ISO o europeo** (la semana comienza el lunes)
- **2 = Calendario islámico** (la semana comienza el sábado)

El valor predeterminado es **0**.

Ejemplos

- `altova:weeknumber-from-dateTime(xs:dateTime("2014-03-23T00:00:00"), 0)` devuelve `13`
- `altova:weeknumber-from-dateTime(xs:dateTime("2014-03-23T00:00:00"), 1)` devuelve `12`
- `altova:weeknumber-from-dateTime(xs:dateTime("2014-03-23T00:00:00"), 2)` devuelve `13`
- `altova:weeknumber-from-dateTime(xs:dateTime("2014-03-23T00:00:00") )` devuelve `13`

El día de `dateTime` de los ejemplos anteriores (`2014-03-23T00:00:00`) es un domingo. Por tanto, en este caso, el calendario estadounidense y el islámico van una semana por delante del calendario europeo.

[Subir 501]

Generar tipos de datos de fecha, hora y duración a partir de sus componentes léxicos XP3.1 XQ3.1

Estas funciones toman los componentes léxicos de los tipos de datos `xs:date`, `xs:time` y `xs:duration` como argumentos de entrada y los combinan para generar el tipo de datos correspondiente.

build-date [altova:]

altova:build-date(Año *as xs:integer*, **Mes** *as xs:integer*, **Fecha** *as xs:integer***)** como **xs:date** XP3.1 XQ3.1

Los argumentos son el año, el mes y la fecha respectivamente. Se combinan para generar un valor de tipo `xs:date`. Los valores de los enteros deben estar en el intervalo de esa fecha en particular. Por ejemplo, el

segundo argumento (para el mes) no puede ser mayor que `12`.

Ejemplos

- **altova:build-date**(2014, 2, 03) devuelve 2014-02-03

build-time [altova:]

altova:build-time(Horas *as xs:integer*, **Minutos** *as xs:integer*, **Segundos** *as xs:integer*) **como xs:time** XP3.1 XQ3.1

El primer, segundo y tercer argumentos son la hora (`0` - `23`), los minutos (`0` - `59`) y los segundos (`0` - `59`) respectivamente. Se combinan para generar un valor de tipo `xs:time`. Los valores de los enteros deben estar dentro del intervalo correcto de esa parte de tiempo concreta. Por ejemplo, el segundo argumento (`Minutos`) no puede ser mayor que `59`. Para añadir la parte de uso horario al valor, use la firma que aparece más abajo.

Ejemplos

- **altova:build-time**(23, 4, 57) devuelve 23:04:57

altova:build-time(Horas *como xs:integer*, **Minutos** *como xs:integer*, **Segundos** *as xs:integer*, **TimeZone** *como xs:string*) **como xs:time** XP3.1 XQ3.1

El primer, segundo y tercer argumentos son la hora (`0` - `23`), los minutos (`0` - `59`) y los segundos (`0` - `59`) respectivamente. El cuarto argumento es una cadena de texto que indica la parte del valor de la zona horaria. Este cuarto argumento se combina para generar un valor de tipo `xs:time`. Los valores de los enteros deben estar dentro del intervalo correcto de esa parte de tiempo concreta. Por ejemplo, el segundo argumento (`Minutos`) no puede ser mayor que `59`.

Ejemplos

- **altova:build-time**(23, 4, 57, '+1') devuelve 23:04:57+01:00

build-duration [altova:]

altova:build-duration(Años *as xs:integer*, **Meses** *as xs:integer*) como **xs:yearMonthDuration** XP3.1 XQ3.1

Toma dos argumentos para generar un valor de tipo `xs:yearMonthDuration`. El primer argumento da la parte `Years` del valor de duración, mientras que el segundo da la parte `Months`. Si el segundo (`Months`) es mayor o igual que `12`, el entero se divide por 12. El cociente se añade al primer argumento para aportar la parte `Years` del valor de duración, mientras que el resto (de la división) da la parte `Months`. Para generar una duración de tipo `xs:dayTimeDuration`, consulte la firma siguiente.

Ejemplos

- **altova:build-duration**(2, 10) devuelve P2Y10M
- **altova:build-duration**(14, 27) devuelve P16Y3M
- **altova:build-duration**(2, 24) devuelve P4Y

altova:build-duration(Días *as xs:integer*, **Horas** *as xs:integer*, **Minutos** *as xs:integer*, **Segundos** *as xs:integer*) como **xs:dayTimeDuration** XP3.1 XQ3.1

Toma cuatro argumentos y los combina para generar un valor de tipo `xs:dayTimeDuration`. El primer argumento da la parte `Days` del valor de duración, el segundo, el tercero y el cuarto dan las partes `Hours`,

`Minutes` y `Seconds` respectivamente. Los tres argumentos de tiempo se convierten a un valor equivalente en cuanto a la unidad mayor siguiente y el resultado se utiliza para calcular el valor total de la duración. Por ejemplo, 72 segundos se convierte en `1M+12S` (1 minuto y 12 segundos) y este valor se usa para calcular el valor total de la duración. Para generar una duración de tipo `xs:yearMonthDuration`, consulte la firma anterior.

Ejemplos

- **`altova:build-duration`**`(2, 10, 3, 56)` devuelve `P2DT10H3M56S`
- **`altova:build-duration`**`(1, 0, 100, 0)` devuelve `P1DT1H40M`
- **`altova:build-duration`**`(1, 0, 0, 3600)` devuelve `P1DT1H`

[Subir 501]

Construir tipos de datos date, dateTime y time a partir de una cadena de entrada XP2 XQ1 XP3.1 XQ3.1

Estas funciones toman cadenas como argumentos y construyen tipos de datos `xs:date`, `xs:dateTime` o `xs:time`. La cadena de entrada se analiza para los componentes del tipo de datos en función del argumento patrón dado.

parse-date [altova:]

`altova:parse-date(Fecha` *como xs:string*`, `**`PatrónFecha`** *como xs:string*`)` como **`xs:date`** XP2 XQ1 XP3.1 XQ3.1

Devuelve la cadena de entrada **`Fecha`** como valor **`xs:date`**. El segundo argumento (**`PatrónFecha`**) indica el patrón (secuencia de componentes) de la cadena de entrada. El argumento **`PatrónFecha`** se describe con los especificadores que aparecen a continuación y con cualquier separador de componentes (consulte los ejemplos más abajo).

`D`	Día
`M`	Mes
`Y`	Año

El patrón **`PatrónFecha`** debe coincidir con el patrón de **`Fecha`**. Como el resultado es de tipo **`xs:date`**, el resultado siempre tendrá el formato léxico **`YYYY-MM-DD`**.

Ejemplos

- **`altova:parse-date`**`(xs:string("09-12-2014"), "[D]-[M]-[Y]")` devuelve `2014-12-09`
- **`altova:parse-date`**`(xs:string("09-12-2014"), "[M]-[D]-[Y]")` devuelve `2014-09-12`
- **`altova:parse-date`**`("06/03/2014", "[M]/[D]/[Y]")` devuelve `2014-06-03`
- **`altova:parse-date`**`("06 03 2014", "[M] [D] [Y]")` devuelve `2014-06-03`
- **`altova:parse-date`**`("6 3 2014", "[M] [D] [Y]")` devuelve `2014-06-03`

parse-dateTime [altova:]

`altova:parse-dateTime(FechaHora` *como xs:string*`, `**`PatrónFechaHora`** *como xs:string*`)` como **`xs:dateTime`** XP2 XQ1 XP3.1 XQ3.1

Devuelve la cadena de entrada **FechaHora** como valor **xs:dateTime**. El segundo argumento (**PatrónFechaHora**) indica el patrón (secuencia de componentes) de la cadena de entrada. El argumento **PatrónFechaHora** se describe con los especificadores que aparecen a continuación y con cualquier separador de componentes (consulte los ejemplos más abajo).

D	Día
M	Mes
Y	Año
H	Hora
m	minutos
s	segundos

El patrón **PatrónFechaHora** debe coincidir con el patrón de **FechaHora**. Como el resultado es de tipo **xs:dateTime**, el resultado siempre tendrá el formato léxico **YYYY-MM-DDTHH:mm:ss**.

Ejemplos

- **altova:parse-dateTime**(xs:string("09-12-2014 13:56:24"), "[M]-[D]-[Y] [H]:[m]:[s]") devuelve 2014-09-12T13:56:24
- **altova:parse-dateTime**("time=13:56:24; date=09-12-2014", "time=[H]:[m]:[s]; date=[D]-[M]-[Y]") devuelve 2014-12-09T13:56:24

parse-time [altova:]

altova:parse-time(Hora *como xs:string*, **PatrónHora** *como xs:string***)** como **xs:time** XP2 XQ1 XP3.1 XQ3.1

Devuelve la cadena de entrada **Hora** como valor **xs:time**. El segundo argumento (**PatrónHora**) indica el patrón (secuencia de componentes) de la cadena de entrada. El argumento **PatrónHora** se describe con los especificadores que aparecen a continuación y con cualquier separador de componentes (consulte los ejemplos más abajo).

H	Hora
m	minutos
s	segundos

El patrón **PatrónHora** debe coincidir con el patrón de **Hora**. Como el resultado es de tipo **xs:time**, el resultado siempre tendrá el formato léxico **HH:mm:ss**.

Ejemplos

- **altova:parse-time**(xs:string("13:56:24"), "[H]:[m]:[s]") devuelve 13:56:24
- **altova:parse-time**("13-56-24", "[H]-[m]") devuelve 13:56:00
- **altova:parse-time**("time=13h56m24s", "time=[H]h[m]m[s]s") devuelve 13:56:24
- **altova:parse-time**("time=24s56m13h", "time=[s]s[m]m[H]h") devuelve 13:56:24

[Subir 501]

Funciones para calcular la edad XP3.1 XQ3.1

Estas funciones devuelven la edad que se calcula obteniendo la diferencia (i) entre la fecha del argumento de entrada y la fecha actual o (ii) entre las fechas de los dos argumentos de entrada. La función **age** devuelve la edad en años, mientras que la función **age-details** devuelve la edad en forma de una secuencia de tres enteros (años, meses y días).

age [altova:]

altova:age(FechaInicio *as xs:date***)** como **xs:integer** XP3.1 XQ3.1
Devuelve un entero que es la edad *en años* de algún objeto, contando a partir de la fecha de inicio dada como argumento y hasta la fecha actual (tomada del reloj del sistema). Si el argumento de entrada es un año o más después que la fecha actual, el valor devuelto será negativo.

Ejemplos

Si la fecha actual es **2014-01-15**:

- `altova:age(xs:date("2013-01-15"))` devuelve `1`
- `altova:age(xs:date("2013-01-16"))` devuelve `0`
- `altova:age(xs:date("2015-01-15"))` devuelve `-1`
- `altova:age(xs:date("2015-01-14"))` devuelve `0`

altova:age(FechaInicio *as xs:date*, **FechaFinal** *as xs:date***)** como **xs:integer** XP3.1 XQ3.1
Devuelve un entero que es la edad *en años* de algún objeto, contando a partir de la fecha de inicio dada como primer argumento y hasta la fecha dada como segundo argumento. El valor devuelto será negativo si el primer argumento es un año o más después que el segundo argumento.

Ejemplos

- `altova:age(xs:date("2000-01-15"), xs:date("2010-01-15"))` devuelve `10`
- `altova:age(xs:date("2000-01-15"), current-date())` devuelve `14` si la fecha actual es `2014-01-15`
- `altova:age(xs:date("2014-01-15"), xs:date("2010-01-15"))` devuelve `-4`

age-details [altova:]

altova:age-details(FechaEntrada *as xs:date***)** como (**xs:integer)*** XP3.1 XQ3.1
Devuelve tres enteros que son los años, meses y días respectivamente que hay entre la fecha dada como argumento y la fecha actual (tomada del reloj del sistema). La suma del valor devuelto nos da el tiempo total transcurrido entre ambas fechas (entre la fecha dada y la fecha actual). La fecha de entrada puede tener un valor anterior o posterior a la fecha actual, pero esto no se indica en el valor devuelto por medio de un signo negativo o positivo. El valor devuelto siempre es positivo.

Ejemplos

Si la fecha actual es **2014-01-15**:

- `altova:age-details(xs:date("2014-01-16"))` devuelve `(0 0 1)`
- `altova:age-details(xs:date("2014-01-14"))` devuelve `(0 0 1)`
- `altova:age-details(xs:date("2013-01-16"))` devuelve `(1 0 1)`

- **altova:age-details**(current-date()) devuelve (0 0 0

altova:age-details(Fecha1 *as xs:date*, **Fecha2** *as xs:date*) como (**xs:integer**)* XP3.1 XQ3.1
Devuelve tres enteros que son los años, meses y días que hay entre las dos fechas dadas por los argumentos. La suma del valor devuelto nos da el tiempo total transcurrido entre las dos fechas de entrada. Da igual cuál de las dos fechas se da como primer argumento, la más antigua o la más reciente. El valor devuelto no indica si la fecha de entrada es anterior o posterior a la fecha actual. Es decir, el valor devuelto siempre es positivo.

Ejemplos

- **altova:age-details**(xs:date("2014-01-16"), xs:date("2014-01-15")) devuelve (0 0 1)
- **altova:age-details**(xs:date("2014-01-15"), xs:date("2014-01-16")) devuelve (0 0 1)

[Subir 501]

Funciones para calcular el tiempo Unix XP3.1 XQ3.1

El tiempo Unix es una medida de tiempo que se usa en sistemas Unix. Se define como la cantidad de segundos transcurridos desde las 00:00:00 UTC del 1 de enero de 1970. Estas funciones convierten valores **xs:dateTime** en tiempo Unix y viceversa.

dateTime-from-epoch [altova:]

altova:dateTime-from-epoch(Epoch *como xs:decimal* **como xs:dateTime** XP3.1 XQ3.1
El tiempo Unix es una medida de tiempo que se usa en sistemas Unix. Se define como la cantidad de segundos transcurridos desde las 00:00:00 UTC del 1 de enero de 1970. La función **dateTime-from-epoch** devuelve el equivalente en **xs:dateTime** de un instante de tiempo Unix, lo ajusta a la zona horaria local e incluye la información de esa zona horaria en el resultado.

La función toma un argumento xs:decimal y devuelve un valor xs:dateTime que incluye una parte **TZ**, que indica la zona horaria. Para obtener el resultado se calcula el equivalente en **dateTime** UTC del instante de tiempo Unix y se añade a la zona horaria local (que se obtiene del reloj del sistema). Por ejemplo, si la función se ejecuta en un equipo cuya configuración sitúa en una zona horaria de +01:00 (con respecto a UTC), una vez se ha calculado el equivalente en **dateTime** se le añade una hora al resultado. La información de la zona horaria, que es una parte léxica opcional del resultado de **xs:dateTime**, también se incluye en el resultado **dateTime**. Compare este resultado con el de **dateTime-from-epoch-no-TZ** y consulte la función **epoch-from-dateTime**.

Ejemplos

La zona horaria local de los ejemplos siguientes es UTC +01:00. En consecuencia, el equivalente en **dateTime** UTC del instante de tiempo Unix indicado aumentará en una hora. La zona horaria se indica en el resultado.

- **altova:dateTime-from-epoch**(34) devuelve 1970-01-01T01:00:34+01:00
- **altova:dateTime-from-epoch**(62) devuelve 1970-01-01T01:01:02+01:00

dateTime-from-epoch-no-TZ [altova:]

altova:dateTime-from-epoch-no-TZ(Epoch *como xs:decimal* **como xs:dateTime** XP3.1 XQ3.1
El tiempo Unix es una medida de tiempo que se usa en sistemas Unix. Se define como la cantidad de segundos transcurridos desde las 00:00:00 UTC del 1 de enero de 1970. La función **dateTime-from-epoch-no-TZ** devuelve el equivalente en **xs:dateTime** de un instante de tiempo Unix y lo ajusta a la zona horaria local pero no incluye la información de esa zona horaria en el resultado.

La función toma un argumento `xs:decimal` y devuelve un valor `xs:dateTime` que no incluye la parte **TZ**, que indica la zona horaria. Para obtener el resultado se calcula el equivalente en **dateTime** UTC del instante de tiempo Unix y se añade a la zona horaria local (que se obtiene del reloj del sistema). Por ejemplo, si la función se ejecuta en un equipo cuya configuración sitúa en una zona horaria de +01:00 (con respecto a UTC), una vez se ha calculado el equivalente en **dateTime** se le añade una hora al resultado. La información de la zona horaria, que es una parte léxica opcional del resultado de **xs:dateTime**, no se incluye en el resultado **dateTime**. Compare este resultado con el de **dateTime-from-epoch** y consulte la función **epoch-from-dateTime**.

Ejemplos

La zona horaria local de los ejemplos siguientes es UTC +01:00. En consecuencia, el equivalente en **dateTime** UTC del instante de tiempo Unix indicado aumentará en una hora. La zona horaria no se indica en el resultado.

- **altova:dateTime-from-epoch**(34) devuelve 1970-01-01T01:00:34
- **altova:dateTime-from-epoch**(62) devuelve 1970-01-01T01:01:02

epoch-from-dateTime [altova:]

altova:epoch-from-dateTime(dateTimeValue *como xs:dateTime*) **como xs:decimal** XP3.1 XQ3.1
El tiempo Unix es una medida de tiempo que se usa en sistemas Unix. Se define como la cantidad de segundos transcurridos desde las 00:00:00 UTC del 1 de enero de 1970. La función **epoch-from-dateTime** devuelve el equivalente en tiempo Unix del valor **xs:dateTime** que se indica en el argumento de la función. Tenga en cuenta que puede que deba generar de forma explícita el valor **xs:dateTime**. El valor **xs:dateTime** puede o no contener la parte opcional **TZ**, que indica la zona horaria.

Tanto si se indica la parte de la zona horaria como parte del argumento como si no, la diferencia que esta indica se obtiene del reloj del sistema y se resta al argumento **dateTimeValue** indicado. El resultado es el tiempo UTC a partir del cual se calcula el equivalente en tiempo Unix. Por ejemplo, si la función se ejecuta en un equipo cuya configuración sitúa en una zona horaria de +01:00 (con respecto a UTC), se resta una hora al valor **dateTimeValue** indicado antes de calcular el valor en tiempo Unix. Consulte también la función **dateTime-from-epoch**.

Ejemplos

La zona horaria local de los ejemplos siguientes es UTC +01:00. En consecuencia, se le restará una hora al valor **dateTime** indicado antes de calcular el tiempo Unix.

- **altova:epoch-from-dateTime**(xs:dateTime("1970-01-01T01:00:34+01:00")) devuelve 34
- **altova:epoch-from-dateTime**(xs:dateTime("1970-01-01T01:00:34")) devuelve 34
- **altova:epoch-from-dateTime**(xs:dateTime("2021-04-01T11:22:33")) devuelve 1617272553

[Subir 501]

12.2.2.1.3 Funciones XPath/XQuery: Geoubicación

Las funciones de extensión XPath/XQuery de geoubicación son compatibles con la versión actual de MapForce y se pueden utilizar en (i) expresiones XPath en contextos XSLT o (ii) expresiones XQuery en documentos XQuery.

Nota sobre el nombre de las funciones y lenguajes

Puede utilizar todas las funciones de extensión de Altova en sus expresiones XPath/XQuery. Con ellas conseguirá funciones adicionales no disponibles en la biblioteca de funciones estándar de XPath, XQuery y XSLT. Las funciones de extensión de Altova están en el **espacio de nombres** `http://www.altova.com/xslt-extensions` y en esta sección se presentan con el prefijo, que se supone estará enlazado al espacio de nombres señalado. Tenga en cuenta que en futuras versiones del producto algunas funciones pueden dejar de ser compatibles o su comportamiento puede cambiar. Por tanto, consulte siempre la documentación del producto para conocer el funcionamiento de estas funciones en cada versión del producto.

Funciones XPath (en expresiones XPath en XSLT):	XP1 XP2 XP3.1.1
Funciones XSLT (en expresiones XPath en XSLT):	XSLT1 XSLT2 XSLT3
Funciones XQuery (en expresiones XQuery en XQuery):	XQ1 XQ3.1

▼ format-geolocation [altova:]

altova:format-geolocation(Latitude *como xs:decimal*, **Longitude** *como xs:decimal*, **GeolocationOutputStringFormat** *como xs:integer*) como **xs:string** XP3.1 XQ3.1

Toma la latitud y la longitud como los dos primeros argumentos y da como resultado la geoubicación como cadena. El tercer argumento, **GeolocationOutputStringFormat**, es el formato de la cadena de resultado de la geoubicación: usa valores enteros del 1 al 4 para identificar el formato de la cadena de resultado (consulte más abajo "*Formatos de la cadena de resultado geoubicación*"). Los valores de latitud oscilan entre `+90` y `-90 (N a S)`. Los valores de longitud oscilan entre `+180` y `-180 (E a O)`.

Nota: la función `image-exif-data` y el atributo de metadatos Exif se pueden usar para suministrar las cadenas de entrada.

- *Ejemplos*
  - **altova:format-geolocation**(33.33, -22.22, 4) devuelve el xs:string "33.33 -22.22"
  - **altova:format-geolocation**(33.33, -22.22, 2) devuelve el xs:string "33.33N 22.22W"
  - **altova:format-geolocation**(-33.33, 22.22, 2) devuelve el xs:string "33.33S 22.22E"
  - **altova:format-geolocation**(33.33, -22.22, 1) devuelve el xs:string "33°19'48.00"S 22° 13'12.00"E"

- *Formato de las cadenas de salida de las geoubicaciones:*

  A la latitud y longitud suministradas se les aplica un formato de salida de los que se indican más

abajo. El formato deseado se identifica con un identificador comprendido entre `1` y `4`. Los valores de latitud pueden estar comprendidos entre `+90` y `-90` (`N` a `S`). Los valores de longitud pueden estar comprendidos entre `+180` y `-180` (`E` a `W`).

1
Grados, minutos y segundos decimales + orientación como sufijo (`N/S, E/W`) **`D°M'S.SS"N/S  D°M'S.SS"E/W`** *Ejemplo*: **`33°55'11.11"N  22°44'66.66"W`**

2
Grados decimales + orientación como sufijo (`N/S, E/W`) **`D.DDN/S  D.DDE/W`** *Ejemplo*: **`33.33N  22.22W`**

3
Grados, minutos y segundos decimales + prefijo (`+/-`). El signo + para (**`N/E`**) es opcional **`+/-D°M'S.SS"  +/-D°M'S.SS"`** *Ejemplo*: **`33°55'11.11"  -22°44'66.66"`**

4
Grados decimales + prefijo (`+/-`). El signo + para (**`N/E`**) es opcional **`+/-D.DD  +/-D.DD`** *Ejemplo*: **`33.33 -22.22`**

Atributo Exif de Altova: Geolocation

El motor XPath/XQuery de Altova genera el atributo personalizado **`Geolocation`** a partir de las etiquetas de metadatos Exif estándar. Este atributo es una concatenación de cuatro etiquetas Exif (`GPSLatitude`, `GPSLatitudeRef`, `GPSLongitude`, `GPSLongitudeRef`) seguidas de unidades:

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

parse-geolocation [altova:]

`altova:parse-geolocation(CadenaEntradaGeoubicación` *como xs:string*`)` como **`xs:decimal+`**
`XP3.1` **`XQ3.1`**
Analiza el argumento `CadenaEntradaGeoubicación` y devuelve la latitud y la longitud (en ese orden) de la geoubicación en forma de secuencia de dos elementos `xs:decimal`. Más abajo puede ver en qué formatos se puede suministrar la cadena de entrada de la geoubicación.

Nota: la función `image-exif-data` 530 y el atributo `@Geolocation` 530 de los metadatos Exif se pueden

utilizar para suministrar la cadena de entrada de la geoubicación (*ver ejemplos*).

- *Ejemplos*

  - `altova:parse-geolocation("33.33  -22.22")` devuelve la secuencia de dos `xs:decimals` `(33.33, 22.22)`
  - `altova:parse-geolocation("48°51'29.6""N  24°17'40.2""")` devuelve la secuencia de dos `xs:decimals` `(48.8582222222222, 24.2945)`
  - `altova:parse-geolocation('48°51''29.6"N  24°17''40.2"')` devuelve la secuencia de dos `xs:decimals` `(48.8582222222222, 24.2945)`
  - `altova:parse-geolocation( image-exif-data(//MisImágenes/Imagen20141130.01)/@Geolocation )` devuelve una secuencia de dos `xs:decimals`

- *Formato de las cadenas de entrada de geoubicaciones:*

  La cadena de entrada de la geoubicación debe contener la latitud y la longitud (en ese orden) se paradas por un espacio en blanco. Ambas pueden estar en cualquier formato de los que se indican más abajo y puede combinar formatos distintos. Es decir, la latitud puede estar en un formato y la longitud en otro. Los valores de la latitud deben estar comprendidos entre `+90` y `-90` (`N` a `S`). Los valores de longitud deben estar comprendidos entre `+180` y `-180` (`E` a `W`).

  Nota: Si utiliza comillas simples o dobles para delimitar el argumento de la cadena de entrada, esto dará lugar a un conflicto con las comillas simples o dobles que se utilizan, respectivamente, para indicar los valores de los minutos y los segundos. Si esto ocurre, debe añadir caracteres de escape a las comillas utilizadas para los minutos y segundos (esto se hace duplicando las comillas). En los ejemplos de esta sección, las comillas para delimitar la cadena de entrada está resaltada en amarillo (**"**) mientras los indicadores de unidades de escape están resaltados en azul (**""**).

  - Grados, minutos y segundos decimales + orientación como sufijo (`N/S`, `E/W`)
    `D°M'S.SS"N/S  D°M'S.SS"W/E`
    Ejemplo: **`33°55'11.11"N  22°44'55.25"W`**

  - Grados, minutos y segundos decimales + prefijo (`+/-`). El signo + para (**`N/E`**) es opcional
    `+/-D°M'S.SS"  +/-D°M'S.SS"`
    Ejemplo: **`33°55'11.11"  -22°44'55.25"`**

  - Grados y minutos decimales + orientación como sufijo (`N/S`, `E/W`)
    `D°M.MM'N/S  D°M.MM'W/E`
    Ejemplo: **`33°55.55'N  22°44.44'W`**

  - Grados y minutos decimales + prefijo (`+/-`). El signo + para (**`N/E`**) es opcional
    `+/-D°M.MM'  +/-D°M.MM'`
    Ejemplo: **`+33°55.55'  -22°44.44'`**

  - Grados decimales + orientación como sufijo (`N/S`, `E/W`)
    `D.DDN/S  D.DDW/E`
    Ejemplo: **`33.33N  22.22W`**

  - Grados decimales + prefijo (`+/-`). El signo + para (**`N/S, E/W`**) es opcional
    `+/-D.DD  +/-D.DD`

Ejemplo: **33.33 -22.22**

Ejemplos de combinación de formatos
33.33N -22°44'55.25"
33.33 22°44'55.25"W
33.33 22.45

- *Atributo Exif de Altova: Geolocation*

  El motor XPath/XQuery de Altova genera el atributo personalizado **Geolocation** a partir de las etiquetas de metadatos Exif estándar. Este atributo es una concatenación de cuatro etiquetas Exif (`GPSLatitude`, `GPSLatitudeRef`, `GPSLongitude`, `GPSLongitudeRef`) seguidas de unidades:

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

- geolocation-distance-km [altova:]

altova:geolocation-distance-km(CadenaEntradaGeoubicación-1 *como xs:string*, **CadenaEntradaGeoubicación-2** *como xs:string***)** como **xs:decimal** XP3.1 XQ3.1
Calcula la distancia en km que existe entre dos geoubicaciones. El formato que puede utilizarse para dar las cadenas de entrada aparece más abajo. Los valores de latitud están comprendidos entre `+90` y `-90` (`N` a `S`). Los valores de longitud están comprendidos entre `+180` y `-180` (`E` a `W`).

Nota: la función `image-exif-data` 530 y el atributo de metadatos Exif `@Geolocation` 530 pueden utilizarse para suministrar las cadenas de entrada de geoubicaciones.

- *Ejemplos*
  - **altova:geolocation-distance-km**(`"33.33 -22.22"`, `"48°51'29.6""N 24°17'40.2"""`) devuelve el `xs:decimal` `4183.08132372392`

- *Formato de las cadenas de entrada de geoubicaciones:*

  La cadena de entrada de la geoubicación debe contener la latitud y la longitud (en ese orden) se paradas por un espacio en blanco. Ambas pueden estar en cualquier formato de los que se indican más abajo y puede combinar formatos distintos. Es decir, la latitud puede estar en un formato y la longitud en otro. Los valores de la latitud deben estar comprendidos entre `+90` y `-90` (`N` a `S`). Los valores de longitud deben estar comprendidos entre `+180` y `-180` (`E` a `W`).

  Nota: Si utiliza comillas simples o dobles para delimitar el argumento de la cadena de entrada, esto dará lugar a un conflicto con las comillas simples o dobles que se utilizan, respectivamente, para indicar los valores de los minutos y los segundos. Si esto ocurre, debe añadir caracteres de escape a las comillas utilizadas para los minutos y segundos (esto se hace duplicando las comillas). En los ejemplos de esta sección, las comillas para delimitar la cadena de entrada está resaltada en amarillo (`"`) mientras los indicadores de unidades de escape están resaltados en azul (`""`).

- Grados, minutos y segundos decimales + orientación como sufijo (`N/S`, `E/W`)
  `D°M'S.SS"N/S  D°M'S.SS"W/E`
  Ejemplo: **`33°55'11.11"N  22°44'55.25"W`**

- Grados, minutos y segundos decimales + prefijo (`+/-`). El signo + para (**`N/E`**) es opcional
  `+/-D°M'S.SS"  +/-D°M'S.SS"`
  Ejemplo: **`33°55'11.11"  -22°44'55.25"`**

- Grados y minutos decimales + orientación como sufijo (`N/S`, `E/W`)
  `D°M.MM'N/S  D°M.MM'W/E`
  Ejemplo: **`33°55.55'N  22°44.44'W`**

- Grados y minutos decimales + prefijo (`+/-`). El signo + para (**`N/E`**) es opcional
  `+/-D°M.MM'  +/-D°M.MM'`
  Ejemplo: **`+33°55.55'  -22°44.44'`**

- Grados decimales + orientación como sufijo (`N/S`, `E/W`)
  `D.DDN/S  D.DDW/E`
  Ejemplo: **`33.33N  22.22W`**

- Grados decimales + prefijo (`+/-`). El signo + para (**`N/S, E/W`**) es opcional
  `+/-D.DD  +/-D.DD`
  Ejemplo: **`33.33  -22.22`**

Ejemplos de combinación de formatos
`33.33N  -22°44'55.25"`
`33.33  22°44'55.25"W`
`33.33  22.45`

Atributo Exif de Altova: Geolocation

El motor XPath/XQuery de Altova genera el atributo personalizado **`Geolocation`** a partir de las etiquetas de metadatos Exif estándar. Este atributo es una concatenación de cuatro etiquetas Exif (`GPSLatitude`, `GPSLatitudeRef`, `GPSLongitude`, `GPSLongitudeRef`) seguidas de unidades:

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

geolocation-distance-mi [altova:]

`altova:geolocation-distance-mi(CadenaEntradaGeoubicación-1` *como xs:string*, **`CadenaEntradaGeoubicación-2`** *como xs:string*`)` como **`xs:decimal`** **XP3.1** **XQ3.1**
Calcula la distancia en millas que existe entre dos geoubicaciones. El formato que puede utilizarse para dar las cadenas de entrada aparece más abajo. Los valores de latitud están comprendidos entre `+90` y `-90` (`N` a `S`). Los valores de longitud están comprendidos entre `+180` y `-180` (`E` a `W`).

Nota: la función `image-exif-data` 530 y el atributo de metadatos Exif `@Geolocation` 530 pueden utilizarse

para suministrar las cadenas de entrada de geoubicaciones.

Ejemplos

- **altova:geolocation-distance-mi**("33.33 -22.22", "48°51'29.6""N 24°17'40.2""")
  devuelve el `xs:decimal` `2599.40652340653`

Formato de las cadenas de entrada de geoubicaciones:

La cadena de entrada de la geoubicación debe contener la latitud y la longitud (en ese orden) se paradas por un espacio en blanco. Ambas pueden estar en cualquier formato de los que se indican más abajo y puede combinar formatos distintos. Es decir, la latitud puede estar en un formato y la longitud en otro. Los valores de la latitud deben estar comprendidos entre `+90` y `-90` (`N` a `S`). Los valores de longitud deben estar comprendidos entre `+180` y `-180` (`E` a `W`).

Nota: Si utiliza comillas simples o dobles para delimitar el argumento de la cadena de entrada, esto dará lugar a un conflicto con las comillas simples o dobles que se utilizan, respectivamente, para indicar los valores de los minutos y los segundos. Si esto ocurre, debe añadir caracteres de escape a las comillas utilizadas para los minutos y segundos (esto se hace duplicando las comillas). En los ejemplos de esta sección, las comillas para delimitar la cadena de entrada está resaltada en amarillo (") mientras los indicadores de unidades de escape están resaltados en azul ("").

- Grados, minutos y segundos decimales + orientación como sufijo (`N/S`, `E/W`)
  D°M'S.SS"N/S D°M'S.SS"W/E
  Ejemplo: **33°55'11.11"N 22°44'55.25"W**

- Grados, minutos y segundos decimales + prefijo (`+/-`). El signo + para (**N/E**) es opcional
  +/-D°M'S.SS" +/-D°M'S.SS"
  Ejemplo: **33°55'11.11" -22°44'55.25"**

- Grados y minutos decimales + orientación como sufijo (`N/S`, `E/W`)
  D°M.MM'N/S D°M.MM'W/E
  Ejemplo: **33°55.55'N 22°44.44'W**

- Grados y minutos decimales + prefijo (`+/-`). El signo + para (**N/E**) es opcional
  +/-D°M.MM' +/-D°M.MM'
  Ejemplo: **+33°55.55' -22°44.44'**

- Grados decimales + orientación como sufijo (`N/S`, `E/W`)
  D.DDN/S D.DDW/E
  Ejemplo: **33.33N 22.22W**

- Grados decimales + prefijo (`+/-`). El signo + para (**N/S, E/W**) es opcional
  +/-D.DD +/-D.DD
  Ejemplo: **33.33 -22.22**

Ejemplos de combinación de formatos

33.33N -22°44'55.25"
33.33 22°44'55.25"W
33.33 22.45

Atributo Exif de Altova: Geolocation

El motor XPath/XQuery de Altova genera el atributo personalizado **Geolocation** a partir de las etiquetas de metadatos Exif estándar. Este atributo es una concatenación de cuatro etiquetas Exif (GPSLatitude, GPSLatitudeRef, GPSLongitude, GPSLongitudeRef) seguidas de unidades:

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

altova:geolocations-bounding-rectangle(Geolocations como *xs:sequence*, **GeolocationOutputStringFormat como** *xs:integer*) como **xs:string** XP3.1 XQ3.1
Toma una secuencia de cadenas de texto como primer argumento; cada cadena de esa secuencia es una geoubicación. La función devuelve una secuencia de dos cadenas que son, respectivamente, las coordinadas de geoubicación de la parte superior izquierda e inferior derecha de un rectángulo delimitado que tiene el tamaño exacto para contener las geoubicaciones suministradas en el primer argumento. Más abajo se enumeran los formatos en que se puede dar una cadena de entrada de geoubicación (*véase "Formato de las cadenas de entrada de geoubicaciones"*). Los valores de latitud están comprendidos entre +90 y -90 (N a S). Los valores de longitud están comprendidos entre +180 y -180 (E a W).

El segundo argumento de la función indica el formato de las dos cadenas de geoubicación de la secuencia de salida. El argumento toma un valor entero entre 1 y 4, donde cada valor representa un formato distinto de las cadenas de entrada de geoubicaciones (*véase "Formato de las cadenas de salida de geoubicaciones"*).

Nota: la función image-exif-data 530 y los atributos de metadatos Exif se pueden usar para suministrar las cadenas de entrada.

- *Ejemplos*
  - **altova:geolocations-bounding-rectangle**(("48.2143531 16.3707266", "51.50939 -0.11832"), 1) devuelve la secuencia ("51°30'33.804"N 0°7'5.952"W", "48°12'51.67116"N 16° 22'14.61576"E")
  - **altova:geolocations-bounding-rectangle**(("48.2143531 16.3707266", "51.50939 -0.11832", "42.5584577 -70.8893334"), 4) devuelve la secuencia ("51.50939 -70.8893334", "42.5584577 16.3707266")

- *Formato de las cadenas de entrada de geoubicaciones:*

  La cadena de entrada de la geoubicación debe contener la latitud y la longitud (en ese orden) se paradas por un espacio en blanco. Ambas pueden estar en cualquier formato de los que se indican más abajo y puede combinar formatos distintos. Es decir, la latitud puede estar en un formato y la longitud en otro. Los valores de la latitud deben estar comprendidos entre +90 y -90 (N a S). Los valores de longitud deben estar comprendidos entre +180 y -180 (E a W).

  Nota: Si utiliza comillas simples o dobles para delimitar el argumento de la cadena de entrada, esto dará lugar a un conflicto con las comillas simples o dobles que se utilizan, respectivamente, para indicar los valores de los minutos y los segundos. Si esto ocurre, debe añadir caracteres de escape a las comillas utilizadas para los minutos y segundos (esto se hace duplicando las comillas). En los ejemplos de esta sección, las comillas para delimitar la cadena de entrada está resaltada en amarillo (") mientras los indicadores de unidades de escape están resaltados en

azul (**""**).

- Grados, minutos y segundos decimales + orientación como sufijo (N/S, E/W)
  D°M'S.SS"N/S D°M'S.SS"W/E
  Ejemplo: **33°55'11.11"N 22°44'55.25"W**

- Grados, minutos y segundos decimales + prefijo (+/-). El signo + para (**N/E**) es opcional
  +/-D°M'S.SS" +/-D°M'S.SS"
  Ejemplo: **33°55'11.11" -22°44'55.25"**

- Grados y minutos decimales + orientación como sufijo (N/S, E/W)
  D°M.MM'N/S D°M.MM'W/E
  Ejemplo: **33°55.55'N 22°44.44'W**

- Grados y minutos decimales + prefijo (+/-). El signo + para (**N/E**) es opcional
  +/-D°M.MM' +/-D°M.MM'
  Ejemplo: **+33°55.55' -22°44.44'**

- Grados decimales + orientación como sufijo (N/S, E/W)
  D.DDN/S D.DDW/E
  Ejemplo: **33.33N 22.22W**

- Grados decimales + prefijo (+/-). El signo + para (**N/S, E/W**) es opcional
  +/-D.DD +/-D.DD
  Ejemplo: **33.33 -22.22**

Ejemplos de combinación de formatos
33.33N -22°44'55.25"
33.33 22°44'55.25"W
33.33 22.45

- *Formato de las cadenas de salida de las geoubicaciones:*

A la latitud y longitud suministradas se les aplica un formato de salida de los que se indican más abajo. El formato deseado se identifica con un identificador comprendido entre 1 y 4. Los valores de latitud pueden estar comprendidos entre +90 y -90 (N a S). Los valores de longitud pueden estar comprendidos entre +180 y -180 (E a W).

1
Grados, minutos y segundos decimales + orientación como sufijo (N/S, E/W) **D°M'S.SS"N/S D°M'S.SS"E/W** *Ejemplo*: **33°55'11.11"N 22°44'66.66"W**

2
Grados decimales + orientación como sufijo (N/S, E/W) **D.DDN/S D.DDE/W** *Ejemplo*: **33.33N 22.22W**

3
Grados, minutos y segundos decimales + prefijo (+/-). El signo + para (N/E) es opcional **+/-D°M'S.SS" +/-D°M'S.SS"** *Ejemplo*: **33°55'11.11" -22°44'66.66"**

4
Grados decimales + prefijo (+/-). El signo + para (N/E) es opcional **+/-D.DD +/-D.DD** *Ejemplo*: **33.33 -22.22**

Atributo Exif de Altova: Geolocation

El motor XPath/XQuery de Altova genera el atributo personalizado **Geolocation** a partir de las etiquetas de metadatos Exif estándar. Este atributo es una concatenación de cuatro etiquetas Exif (`GPSLatitude`, `GPSLatitudeRef`, `GPSLongitude`, `GPSLongitudeRef`) seguidas de unidades:

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

geolocation-within-polygon [altova:]

altova:geolocation-within-polygon(Geoubicación *como xs:string*, **((PuntoDePolígono** *como xs:string*)+)) como **xs:boolean** XP3.1 XQ3.1

Determina si **Geoubicación** (primer argumento) está dentro del área poligonal descrita por los argumentos **PuntoDePolígono**. Si los argumentos `PuntoDePolígono` no forman una figura cerrada (la figura se cierra cuando el primer y el último punto son el mismo), entonces el primer punto se añade implícitamente como último punto a fin de cerrar la figura. Todos los argumentos (`Geoubicación` y `PuntoDePolígono+`) se dan como cadenas de entrada de geoubicación (*formatos permitidos más abajo*). Si el argumento `Geoubicación` está dentro del área poligonal, entonces la función devuelve `true()`. De lo contrario, devuelve `false()`. Los valores de latitud están comprendidos entre `+90` y `-90` (`N` a `S`). Los valores de longitud están comprendidos entre `+180` y `-180` (`E` a `W`).

Nota: la función `image-exif-data` 530 y el atributo de metadatos Exif `@Geolocation` 530 pueden utilizarse para suministrar las cadenas de entrada de geoubicaciones.

Ejemplos

- **altova:geolocation-within-polygon(**"33 -22", ("58 -32", "-78 -55", "48 24", "58 -32")**)** devuelve `true()`
- **altova:geolocation-within-polygon(**"33 -22", ("58 -32", "-78 -55", "48 24")**)** devuelve `true()`
- **altova:geolocation-within-polygon(**"33 -22", ("58 -32", "-78 -55", "48°51'29.6""N 24°17'40.2""")**)** devuelve `true()`

Formato de las cadenas de entrada de geoubicaciones:

La cadena de entrada de la geoubicación debe contener la latitud y la longitud (en ese orden) se paradas por un espacio en blanco. Ambas pueden estar en cualquier formato de los que se indican más abajo y puede combinar formatos distintos. Es decir, la latitud puede estar en un formato y la longitud en otro. Los valores de la latitud deben estar comprendidos entre `+90` y `-90` (`N` a `S`). Los valores de longitud deben estar comprendidos entre `+180` y `-180` (`E` a `W`).

Nota: Si utiliza comillas simples o dobles para delimitar el argumento de la cadena de entrada, esto dará lugar a un conflicto con las comillas simples o dobles que se utilizan, respectivamente, para indicar los valores de los minutos y los segundos. Si esto ocurre, debe añadir caracteres de escape a las comillas utilizadas para los minutos y segundos (esto se hace duplicando las comillas). En los ejemplos de esta sección, las comillas para delimitar la cadena de entrada está resaltada en amarillo (**''**) mientras los indicadores de unidades de escape están resaltados en azul (**""**).

- Grados, minutos y segundos decimales + orientación como sufijo (`N/S`, `E/W`)
  D°M'S.SS"N/S D°M'S.SS"W/E
  Ejemplo: **33°55'11.11"N 22°44'55.25"W**

- Grados, minutos y segundos decimales + prefijo (`+/-`). El signo + para (**N/E**) es opcional
  +/-D°M'S.SS" +/-D°M'S.SS"
  Ejemplo: **33°55'11.11" -22°44'55.25"**

- Grados y minutos decimales + orientación como sufijo (`N/S`, `E/W`)
  D°M.MM'N/S D°M.MM'W/E
  Ejemplo: **33°55.55'N 22°44.44'W**

- Grados y minutos decimales + prefijo (`+/-`). El signo + para (**N/E**) es opcional
  +/-D°M.MM' +/-D°M.MM'
  Ejemplo: **+33°55.55' -22°44.44'**

- Grados decimales + orientación como sufijo (`N/S`, `E/W`)
  D.DDN/S D.DDW/E
  Ejemplo: **33.33N 22.22W**

- Grados decimales + prefijo (`+/-`). El signo + para (**N/S, E/W**) es opcional
  +/-D.DD +/-D.DD
  Ejemplo: **33.33 -22.22**

Ejemplos de combinación de formatos

```
33.33N  -22°44'55.25"
33.33  22°44'55.25"W
33.33  22.45
```

Atributo Exif de Altova: Geolocation

El motor XPath/XQuery de Altova genera el atributo personalizado **Geolocation** a partir de las etiquetas de metadatos Exif estándar. Este atributo es una concatenación de cuatro etiquetas Exif (`GPSLatitude`, `GPSLatitudeRef`, `GPSLongitude`, `GPSLongitudeRef`) seguidas de unidades:

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation

33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

▼ geolocation-within-rectangle [altova:]

altova:geolocation-within-rectangle(Geoubicación *como xs:string*, **ÁnguloRectángulo-1** *como xs:string*, **ÁnguloRectángulo-2** *como xs:string*) como **xs:boolean** XP3.1 XQ3.1
Determina si **Geoubicación** (primer argumento) está dentro del rectángulo definido por el segundo y el tercer argumento (**ÁnguloRectángulo-1** y **ÁnguloRectángulo-2**), que indican ángulos opuestos del rectángulo. Todos los argumentos de la función se dan como cadenas de entrada de geoubicación (*formatos permitidos más abajo*). Si el argumento `Geoubicación` está dentro del rectángulo, entonces la función devuelve `true()`. De lo contrario, devuelve `false()`. Los valores de latitud están comprendidos entre `+90` y `-90` (`N` a `S`). Los valores de longitud están comprendidos entre `+180` y `-180` (`E` a `W`).

Nota: la función `image-exif-data` 530 y el atributo de metadatos Exif `@Geolocation` 530 pueden utilizarse para suministrar las cadenas de entrada de geoubicaciones.

▣ *Ejemplos*

- `altova:geolocation-within-rectangle("33 -22", "58 -32", "-48 24")` devuelve `true()`
- `altova:geolocation-within-rectangle("33 -22", "58 -32", "48 24")` devuelve `false()`
- `altova:geolocation-within-rectangle("33 -22", "58 -32", "48°51'29.6""S 24° 17'40.2""")` devuelve `true()`

▣ *Formato de las cadenas de entrada de geoubicaciones:*

La cadena de entrada de la geoubicación debe contener la latitud y la longitud (en ese orden) se paradas por un espacio en blanco. Ambas pueden estar en cualquier formato de los que se indican más abajo y puede combinar formatos distintos. Es decir, la latitud puede estar en un formato y la longitud en otro. Los valores de la latitud deben estar comprendidos entre `+90` y `-90` (`N` a `S`). Los valores de longitud deben estar comprendidos entre `+180` y `-180` (`E` a `W`).

Nota: Si utiliza comillas simples o dobles para delimitar el argumento de la cadena de entrada, esto dará lugar a un conflicto con las comillas simples o dobles que se utilizan, respectivamente, para indicar los valores de los minutos y los segundos. Si esto ocurre, debe añadir caracteres de escape a las comillas utilizadas para los minutos y segundos (esto se hace duplicando las comillas). En los ejemplos de esta sección, las comillas para delimitar la cadena de entrada está resaltada en amarillo (`"`) mientras los indicadores de unidades de escape están resaltados en azul (`""`).

- Grados, minutos y segundos decimales + orientación como sufijo (`N/S`, `E/W`)
  `D°M'S.SS"N/S D°M'S.SS"W/E`
  Ejemplo: **`33°55'11.11"N 22°44'55.25"W`**

- Grados, minutos y segundos decimales + prefijo (`+/-`). El signo + para (**`N/E`**) es opcional
  `+/-D°M'S.SS" +/-D°M'S.SS"`
  Ejemplo: **`33°55'11.11" -22°44'55.25"`**

- Grados y minutos decimales + orientación como sufijo (`N/S`, `E/W`)
  `D°M.MM'N/S  D°M.MM'W/E`
  Ejemplo: **`33°55.55'N  22°44.44'W`**

- Grados y minutos decimales + prefijo (`+/-`). El signo + para (**`N/E`**) es opcional
  `+/-D°M.MM'  +/-D°M.MM'`
  Ejemplo: **`+33°55.55'  -22°44.44'`**

- Grados decimales + orientación como sufijo (`N/S`, `E/W`)
  `D.DDN/S  D.DDW/E`
  Ejemplo: **`33.33N  22.22W`**

- Grados decimales + prefijo (`+/-`). El signo + para (**`N/S, E/W`**) es opcional
  `+/-D.DD  +/-D.DD`
  Ejemplo: **`33.33  -22.22`**

Ejemplos de combinación de formatos
`33.33N  -22°44'55.25"`
`33.33  22°44'55.25"W`
`33.33  22.45`

Atributo Exif de Altova: Geolocation

El motor XPath/XQuery de Altova genera el atributo personalizado **`Geolocation`** a partir de las etiquetas de metadatos Exif estándar. Este atributo es una concatenación de cuatro etiquetas Exif (`GPSLatitude`, `GPSLatitudeRef`, `GPSLongitude`, `GPSLongitudeRef`) seguidas de unidades:

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

[Subir 519]

12.2.2.1.4 Funciones XPath/XQuery: Imágenes

Las funciones de extensión XPath/XQuery para trabajar con imágenes son compatibles con la versión actual de MapForce y se pueden utilizar en (i) expresiones XPath en contextos XSLT o (ii) expresiones XQuery en documentos XQuery.

Nota sobre el nombre de las funciones y lenguajes

Puede utilizar todas las funciones de extensión de Altova en sus expresiones XPath/XQuery. Con ellas conseguirá funciones adicionales no disponibles en la biblioteca de funciones estándar de XPath, XQuery y XSLT. Las funciones de extensión de Altova están en el **espacio de nombres** **`http://www.altova.com/xslt-extensions`** y en esta sección se presentan con el prefijo, que se supone estará enlazado al espacio de nombres señalado. Tenga en cuenta que en futuras versiones del producto

algunas funciones pueden dejar de ser compatibles o su comportamiento puede cambiar. Por tanto, consulte siempre la documentación del producto para conocer el funcionamiento de estas funciones en cada versión del producto.

Funciones XPath (en expresiones XPath en XSLT):	XP1 XP2 XP3.1.1
Funciones XSLT (en expresiones XPath en XSLT):	XSLT1 XSLT2 XSLT3
Funciones XQuery (en expresiones XQuery en XQuery):	XQ1 XQ3.1

▼ suggested-image-file-extension [altova:]

altova:suggested-image-file-extension(CadenaBase64 *como string***)** como **string?** XP3.1 XQ3.1
Toma la codificación base64 de un archivo de imagen como argumento y devuelve la extensión de archivo de la imagen registrada en la codificación base64 de la imagen. El valor devuelto es una sugerencia basada en la información sobre el tipo de imagen disponible en la codificación. Si esta información no está disponible, entonces devuelve una cadena vacía. Esta función es muy práctica a la hora de guardar una imagen base64 como archivo y recuperar de forma dinámica una extensión de archivo adecuada.

▣ *Ejemplos*

- `altova:suggested-image-file-extension(/MisImágenes/TeléfonoMóvil/Imagen20141130.01)` devuelve `'jpg'`
- `altova:suggested-image-file-extension($XML1/Personal/Persona/@photo)` devuelve `''`

En los ejemplos anteriores, se da por hecho que los nodos suministrados como argumento de la función contienen una imagen codificada en base64. El primer ejemplo recupera `jpg` como tipo de imagen y como extensión de archivo. En el segundo ejemplo, la codificación base64 dada no ofrece información sobre la extensión del archivo.

▼ image-exif-data [altova:]

altova:image-exif-data(CadenaBinariaBase64 *como string***)** como **element?** XP3.1 XQ3.1
Toma una imagen JPEG codificada en base64 como argumento y devuelve un elemento llamado **Exif** que contiene los metadatos Exif de la imagen. Los metadatos Exif se crean como pares atributo-valor del elemento `Exif`. El nombre de los atributos son las etiquetas de datos Exif encontradas en la codificación base64. La lista de etiquetas Exif aparece más abajo. Si en lo datos Exif hay etiquetas de terceros, estas etiquetas y sus valores también se devuelven en un par atributo-valor. Además de las etiquetas de metadatos Exif estándar (*lista más abajo*), también se generan pares atributo-valor de Altova. Estos atributos Exif de Altova también se enumeran más abajo.

▣ *Ejemplos*

- Para acceder a un atributo, utilice la función de esta manera:
  `image-exif-data(//MisImágenes/Imagen20141130.01)/@GPSLatitude`
  `image-exif-data(//MisImágenes/Imagen20141130.01)/@Geolocation`
- Para acceder a todos los atributos, utilice la función de esta manera:
  `image-exif-data(//MisImágenes/Imagen20141130.01)/@*`

- Para acceder al nombre de todos los atributos, utilice esta expresión:
  `for $i in image-exif-data(//MisImágenes/Imagen20141130.01)/@* return name($i)`
  Esto es muy práctico a la hora de averiguar el nombre de los atributos que devuelve la función.

Atributo Exif de Altova: Geolocation

El motor XPath/XQuery de Altova genera el atributo personalizado **Geolocation** a partir de las etiquetas de metadatos Exif estándar. Este atributo es una concatenación de cuatro etiquetas Exif (`GPSLatitude`, `GPSLatitudeRef`, `GPSLongitude`, `GPSLongitudeRef`) seguidas de unidades:

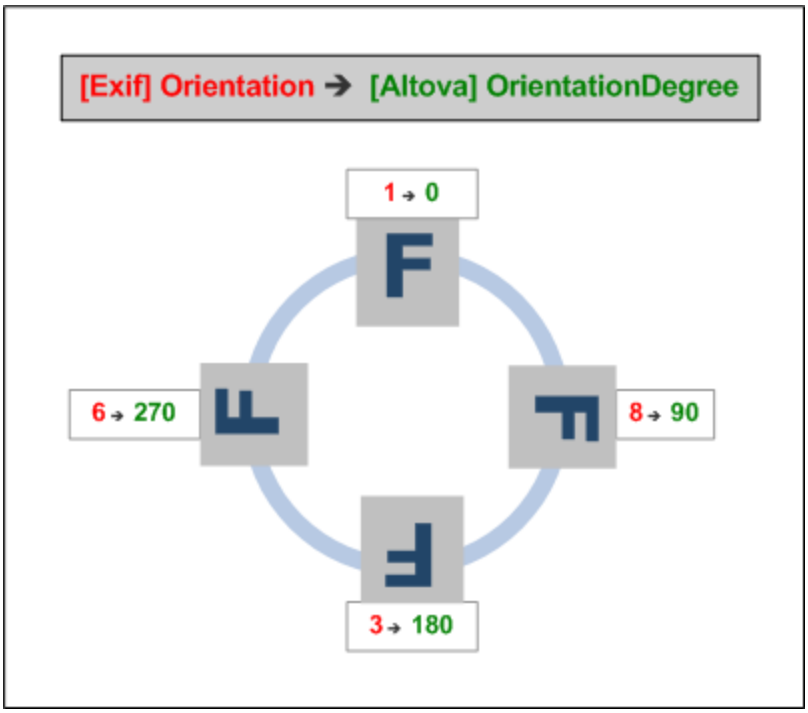
GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

Atributo Exif de Altova: OrientationDegree

El motor XPath/XQuery de Altova genera el atributo personalizado **OrientationDegree** a partir de la etiqueta de metadatos Exif **Orientation**.

Este atributo transforma el valor entero de la etiqueta Exif `Orientation` (`1`, `8`, `3` o `6`) en el correspondiente valor en grados (`0`, `90`, `180`, `270`), tal y como describe el diagrama más abajo.

Debe tener en cuenta que los valores `2`, `4`, `5`, `7` de `Orientation` no se pueden traducir. Estas orientaciones se obtienen invirtiendo la imagen `1` en su eje central vertical para obtener la imagen con un valor de `2` e invirtiendo después esta imagen por pasos de 90 grados en el sentido de las agujas del reloj para obtener los valores de `7`, `4` y `5`, respectivamente.



- *Lista de etiquetas Exif estándar*
  - ImageWidth
  - ImageLength
  - BitsPerSample
  - Compression
  - PhotometricInterpretation
  - Orientation
  - SamplesPerPixel
  - PlanarConfiguration
  - YCbCrSubSampling
  - YCbCrPositioning
  - XResolution
  - YResolution
  - ResolutionUnit
  - StripOffsets
  - RowsPerStrip
  - StripByteCounts
  - JPEGInterchangeFormat
  - JPEGInterchangeFormatLength
  - TransferFunction
  - WhitePoint
  - PrimaryChromaticities
  - YCbCrCoefficients
  - ReferenceBlackWhite
  - DateTime
  - ImageDescription
  - Make

- Model
- Software
- Artist
- Copyright

- ExifVersion
- FlashpixVersion
- ColorSpace
- ComponentsConfiguration
- CompressedBitsPerPixel
- PixelXDimension
- PixelYDimension
- MakerNote
- UserComment
- RelatedSoundFile
- DateTimeOriginal
- DateTimeDigitized
- SubSecTime
- SubSecTimeOriginal
- SubSecTimeDigitized
- ExposureTime
- FNumber
- ExposureProgram
- SpectralSensitivity
- ISOSpeedRatings
- OECF
- ShutterSpeedValue
- ApertureValue
- BrightnessValue
- ExposureBiasValue
- MaxApertureValue
- SubjectDistance
- MeteringMode
- LightSource
- Flash
- FocalLength
- SubjectArea
- FlashEnergy
- SpatialFrequencyResponse
- FocalPlaneXResolution
- FocalPlaneYResolution
- FocalPlaneResolutionUnit
- SubjectLocation
- ExposureIndex
- SensingMethod
- FileSource
- SceneType
- CFAPattern
- CustomRendered
- ExposureMode
- WhiteBalance
- DigitalZoomRatio
- FocalLengthIn35mmFilm
- SceneCaptureType

- GainControl
- Contrast
- Saturation
- Sharpness
- DeviceSettingDescription
- SubjectDistanceRange
- ImageUniqueID

- GPSVersionID
- GPSLatitudeRef
- GPSLatitude
- GPSLongitudeRef
- GPSLongitude
- GPSAltitudeRef
- GPSAltitude
- GPSTimeStamp
- GPSSatellites
- GPSStatus
- GPSMeasureMode
- GPSDOP
- GPSSpeedRef
- GPSSpeed
- GPSTrackRef
- GPSTrack
- GPSImgDirectionRef
- GPSImgDirection
- GPSMapDatum
- GPSDestLatitudeRef
- GPSDestLatitude
- GPSDestLongitudeRef
- GPSDestLongitude
- GPSDestBearingRef
- GPSDestBearing
- GPSDestDistanceRef
- GPSDestDistance
- GPSProcessingMethod
- GPSAreaInformation
- GPSDateStamp
- GPSDifferential

[Subir 530]

12.2.2.1.5 Funciones XPath/XQuery: Numéricas

Las funciones de extensión numéricas de Altova pueden utilizarse en expresiones XPath y XQuery y ofrecen funciones adicionales para el procesamiento de datos. Estas funciones se pueden usar con los motores **XPath 3.0** y **XQuery 3.0** de Altova. Están disponibles en contextos XPath/XQuery.

Nota sobre el nombre de las funciones y lenguajes

Puede utilizar todas las funciones de extensión de Altova en sus expresiones XPath/XQuery. Con ellas conseguirá funciones adicionales no disponibles en la biblioteca de funciones estándar de XPath, XQuery y XSLT. Las funciones de extensión de Altova están en el **espacio de nombres** **`http://www.altova.com/xslt-extensions`** y en esta sección se presentan con el prefijo, que se supone estará enlazado al espacio de nombres señalado. Tenga en cuenta que en futuras versiones del producto algunas funciones pueden dejar de ser compatibles o su comportamiento puede cambiar. Por tanto, consulte siempre la documentación del producto para conocer el funcionamiento de estas funciones en cada versión del producto.

Funciones XPath (en expresiones XPath en XSLT):	XP1 XP2 XP3.1.1
Funciones XSLT (en expresiones XPath en XSLT):	XSLT1 XSLT2 XSLT3
Funciones XQuery (en expresiones XQuery en XQuery):	XQ1 XQ3.1

Funciones de numeración automática

▼ generate-auto-number [altova:]

`altova:generate-auto-number(ID` *como xs:string*`,` **`EmpiezaPor`** *como xs:double*`,` **`Incremento`** *como xs:double*`,` **`RestaurarAlCambiar`** *como xs:string*`)` como `xs:integer` XP1 XP2 XQ1 XP3.1 XQ3.1

Genera un número cada vez que se llama a la función. El primer número, que se genera cuando se llama a la función por primera vez, viene dado por el argumento `EmpiezaPor`. Cada llamada posterior genera un número nuevo, que se incrementa en función del valor especificado en el argumento `Incremento`. De hecho, la función `generate-auto-number` crea un contador llamado como indique el argumento `ID` y este contador se incrementa cada vez que se llama a la función. Si el valor del argumento `RestaurarAlCambiar` cambia con respecto al valor que tenía en la llamada anterior, entonces el valor del número que se debe generar se restablece con el valor de `EmpiezaPor`. También puede restablecer la numeración automática con la función `altova:reset-auto-number`.

▣ *Ejemplo*

- **`altova:generate-auto-number`**`("ChapterNumber", 1, 1, "SomeString")`

Devuelve un número cada vez que se llama a la función, empezando por `1` y con un incremento de `1` con cada llamada a función. Si el cuarto argumento continúa siendo `"SomeString"` en las llamadas posteriores, el incremento continuará. Cuando cambie el valor del cuarto argumento, se restaura el valor 1 del contador (llamado `ChapterNumber`). El valor de `ChapterNumber` también se puede restaurar llamando a la función `altova:reset-auto-number("ChapterNumber")`.

▼ reset-auto-number [altova:]

`altova:reset-auto-number(ID` *como xs:string*`)` XP1 XP2 XQ1 XP3.1 XQ3.1

Esta función restaura el número del contador de numeración automática especificado en el argumento `ID`. El número se reemplaza con el número indicado en el argumento `EmpiezaPor` de la función `altova:generate-auto-number` que creó el contador especificado en el argumento `ID`.

▣ *Ejemplos*

- **`altova:reset-auto-number`**`("ChapterNumber")` restablece el número del contador de

numeración automática llamado `ChapterNumber` que se creó con la función `altova:generate-auto-number`. El número se reemplaza con el valor del argumento `EmpiezaPor` de la función `altova:generate-auto-number` que creó `ChapterNumber`.

[Subir 535]

Funciones numéricas

▼ hex-string-to-integer [altova:]

`altova:hex-string-to-integer(CadenaHex` *`as xs:string`***`)`** como **`xs:integer`** XP3.1 XQ3.1
Toma un argumento de cadena que es el equivalente Base-16 de un entero del sistema decimal (Base-10) y devuelve un entero decimal.

Ejemplos

- **`altova:hex-string-to-integer`**`('1')` devuelve `1`
- **`altova:hex-string-to-integer`**`('9')` devuelve `9`
- **`altova:hex-string-to-integer`**`('A')` devuelve `10`
- **`altova:hex-string-to-integer`**`('B')` devuelve `11`
- **`altova:hex-string-to-integer`**`('F')` devuelve `15`
- **`altova:hex-string-to-integer`**`('G')` devuelve un error
- **`altova:hex-string-to-integer`**`('10')` devuelve `16`
- **`altova:hex-string-to-integer`**`('01')` devuelve `1`
- **`altova:hex-string-to-integer`**`('20')` devuelve `32`
- **`altova:hex-string-to-integer`**`('21')` devuelve `33`
- **`altova:hex-string-to-integer`**`('5A')` devuelve `90`
- **`altova:hex-string-to-integer`**`('USA')` devuelve un error

▼ integer-to-hex-string [altova:]

`altova:integer-to-hex-string(Entero` *`as xs:integer`***`)`** como **`xs:string`** XP3.1 XQ3.1
Toma el argumento **`Entero`** y devuelve su equivalente Base-16 en forma de cadena.

Ejemplos

- **`altova:integer-to-hex-string`**`(1)` devuelve `'1'`
- **`altova:integer-to-hex-string`**`(9)` devuelve `'9'`
- **`altova:integer-to-hex-string`**`(10)` devuelve `'A'`
- **`altova:integer-to-hex-string`**`(11)` devuelve `'B'`
- **`altova:integer-to-hex-string`**`(15)` devuelve `'F'`
- **`altova:integer-to-hex-string`**`(16)` devuelve `'10'`
- **`altova:integer-to-hex-string`**`(32)` devuelve `'20'`
- **`altova:integer-to-hex-string`**`(33)` devuelve `'21'`
- **`altova:integer-to-hex-string`**`(90)` devuelve `'5A'`

[Subir 535]

Funciones de formato numérico

[Subir 535]

12.2.2.1.6 Funciones XPath/XQuery: Esquema

Las funciones de extensión de Altova que enumeramos a continuación devuelven información del esquema. Más adelante verá descripciones de las funciones, junto con (i) ejemplos y (ii) una lista de los componentes del esquema y sus correspondientes propiedades. Estas funciones se pueden usar con los motores de Altova **XPath 3.0** y **XQuery 3.0**, y están disponibles en contextos XPath/XQuery.

Información sobre el esquema proveniente de documentos de esquema
La función **altova:schema** tiene dos argumentos: uno que no tiene argumentos y otro que tiene dos. La función que no tiene argumentos devuelve todo el esquema. A partir de ahí puede navegar por el esquema para encontrar los componentes que necesite. La función con dos argumentos devuelve un tipo concreto de componente al que se identifica por su QName. En ambos casos el valor de retorno es una función. Para ir al componente devuelto debe seleccionar una de sus propiedades. Si esta propiedad es un elemento no atómico (es decir, si es un componente), entonces puede seleccionar también una propiedad de este componente para seguir navegando. Si la propiedad seleccionada sí es un elemento atómico, entonces se devuelve el valor del elemento y no puede seguir navegando.

Nota: en las expresiones XPath de debe importar primero el esquema en el entorno de procesamiento (por ejemplo, XSLT), con la instrucción **xslt:import-schema**. En las expresiones XQuery, el esquema se debe importar de forma explícita.

Información sobre el esquema proveniente de nodos XML
La función **altova:type** envía el nodo de un documento XML y devuelve la información del tipo del modo desde el PSVI (Conjunto de información posterior a la validación de esquemas).

Nota sobre el nombre de las funciones y lenguajes

Puede utilizar todas las funciones de extensión de Altova en sus expresiones XPath/XQuery. Con ellas conseguirá funciones adicionales no disponibles en la biblioteca de funciones estándar de XPath, XQuery y XSLT. Las funciones de extensión de Altova están en el **espacio de nombres** **http://www.altova.com/xslt-extensions** y en esta sección se presentan con el prefijo, que se supone estará enlazado al espacio de nombres señalado. Tenga en cuenta que en futuras versiones del producto algunas funciones pueden dejar de ser compatibles o su comportamiento puede cambiar. Por tanto, consulte siempre la documentación del producto para conocer el funcionamiento de estas funciones en cada versión del producto.

Funciones XPath (en expresiones XPath en XSLT):	XP1 XP2 XP3.1.1
Funciones XSLT (en expresiones XPath en XSLT):	XSLT1 XSLT2 XSLT3
Funciones XQuery (en expresiones XQuery en XQuery):	XQ1 XQ3.1

altova:schema() como (function(xs:string) como item()*)? XP3.1 XQ3.1
Devuelve el componente **schema** al completo. Para navegar por este componente seleccione una de sus propiedades.

- Si esta propiedad es un componente seleccione una de sus propiedades para navegar hasta el siguiente nivel de profundidad. Puede repetir este paso para seguir navegando por el esquema.
- Si el componente es un valor atómico se devuelve este valor y no puede seguir navegando.

Las propiedades del componente **schema** son:

```
"type definitions"
"attribute declarations"
"element declarations"
"attribute group definitions"
"model group definitions"
"notation declarations"
"identity-constraint definitions"
```

Más abajo encontrará las propiedades del resto de tipos de componente.

Nota: en las expresiones XQuery, el esquema se debe importar de forma explícita. En las expresiones XPath debe importar primero el esquema en el entorno de procesamiento, por ejemplo en XSLT con la instrucción **xslt:import**.

Ejemplos

- `import schema "" at "C:\Test\ExpReport.xsd"; for $typedef in altova:schema() ("type definitions")`
  `return $typedef ("name")` devuelve los nombres de todos los tipos simples o complejos del esquema

- `import schema "" at "C:\Test\ExpReport.xsd";`
  `altova:schema() ("type definitions")[1]("name")` devuelve el nombre del primero de los tipos simples o complejos del esquema

Componentes y sus propiedades

Assertion

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Assertion"
test	Registro de propiedades XPath	

Attribute Declaration

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Attribute Declaration"

name	Cadena	Nombre local del atributo
target namespace	Cadena	URI del espacio de nombres del atributo
type definition	Simple Type o Complex Type	
scope	Una función con propiedades ("class":"Scope", "variety": "global" o "local", "parent": el Complex Type o Attribute Group contenedor)	
value constraint	Si está presente, una función con propiedades ("class": "Value Constraint", "variety": "fixed" o "default", "value": atomic value, "lexical form": string. Tenga en cuenta que la propiedad "value" no está disponible para los tipos namespace-sensitive	
inheritable	Booleano	

- Attribute Group Definition

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Attribute Group Definition"
name	Cadena	Nombre local del grupo de atributos
target namespace	Cadena	URI del espacio de nombres del grupo de atributos
attribute uses	Secuencia de (Attribute Use)	
attribute wildcard	Comodín de atributo opcional	

- Attribute Use

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Attribute Use"
required	Booleano	true si el atributo es obligatorio, false si es opcional
value constraint	Véase la declaración de atributos	
inheritable	Booleano	

- Attribute Wildcard

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Wildcard"
namespace constraint	Función con propiedades ("class":	

	"Namespace Constraint", "variety": "any"|"enumeration"|"not", "namespaces": secuencia de elementos xs:anyURI, "disallowed names": lista que contiene QNames y/o las cadenas "defined" y "definedSiblings"	
process contents	Cadena ("strict"|"lax"|"skip")	

- Complex Type

Nombre de la propiedad	**Tipo de la propiedad**	**Valor de la propiedad**
kind	Cadena	"Complex Type"
name	Cadena	Nombre local del tipo (vacío si es anónimo)
target namespace	Cadena	URI del espacio de nombres del tipo (vacío si es anónimo)
base type definition	Definición del Complex Type	
final	Secuencia de cadenas ("restriction"|"extension")	
context	Secuencia vacía (not implemented)	
derivation method	Cadena ("restriction"|"extension")	
abstract	Booleano	
attribute uses	Secuencia de elementos Attribute Use	
attribute wildcard	Comodín de atributo opcional	
content type	Función con propiedades: ("class":"Content Type", "variety":string ("element-only"|"empty"|"mixed"|"simple"), particle: partícula opcional, "open content": función con propiedades ("class":"Open Content", "mode": string ("interleave"|"suffix"), "wildcard": Wildcard), "simple type definition": Simple Type)	
prohibited substitutions	Secuencia de cadenas ("restriction"|"extension")	
assertions	Secuencia de elementos Assertion	

- Element Declaration

Nombre de la propiedad	**Tipo de la propiedad**	**Valor de la propiedad**
kind	Cadena	"Complex Type"
name	Cadena	Nombre local del tipo (vacío si es

		anónimo)
target namespace	Cadena	Namespace URI del tipo (vacío si es anónimo)
type definition	Simple Type o Complex Type	
type table	Función con propiedades ("class":"Type Table", "alternatives": secuencia de elementos Type Alternative, "default type definition": Simple Type o Complex Type)	
scope	Función con propiedades ("class":"Scope", "variety": ("global"|"local"), "parent": Complex Type opcional)	
value constraint	véase Attribute Declaration	
nillable	Booleano	
identity-constraint definitions	Secuencia de restricciones de identidad	
substitution group affiliations	Secuencia de declaraciones de elementos	
substitution group exclusions	Secuencia de cadenas ("restriction"|"extension")	
disallowed substitutions	Secuencia de cadenas ("restriction"|"extension"|"substitution")	
abstract	Booleano	

Element Wildcard

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Wildcard"
namespace constraint	Función con propiedades ("class": "Namespace Constraint", "variety": "any"|"enumeration"|"not", "namespaces": secuencia de xs:anyURI, "disallowed names": lista que contiene QNames y/o las cadenas "defined" y "definedSiblings"	
process contents	Cadena ("strict"|"lax"|"skip")	

Facet

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	El nombre de la faceta, por ejemplo "minLength" o "enumeration"
value	Depende de la faceta	El valor de la faceta

fixed	Booleano	
typed-value	Sólo para facetas de enumeración, Array(xs:anyAtomicType*)	Una matriz que contiene los valores de la enumeración, cada uno de los cuales puede ser una secuencia de valores atómicos. (Nota: para la faceta de enumeración, la propiedad "value" es un secuencia de cadenas, independientemente del tipo)

Identity Constraint

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Identity-Constraint Definition"
name	Cadena	Nombre local de la restricción
target namespace	Cadena	URI del espacio de nombres de la restricción
identity-constraint category	Cadena ("key"|"unique"|"keyRef")	
selector	Registro de propiedades XPath	
fields	Secuencia de registros de propiedades XPath	
referenced key	(Sólo para keyRef): Identity Constraint	La restricción clave correspondiente

Model Group

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Model Group"
compositor	Cadena ("sequence"|"choice"|"all")	
particles	Secuencia de partículas	

Model Group Definition

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Model Group Definition"
name	Cadena	Nombre local del grupo de modelos
target namespace	Cadena	URI del espacio de nombres del grupo de modelos
model group	Model Group	

Notation

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Notation Declaration"
name	Cadena	Nombre local de la notación
target namespace	Cadena	URI del espacio de nombres de la notación
system identifier	anyURI	
public identifier	Cadena	

Particle

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Particle"
min occurs	Número entero	
max occurs	Número entero o cadena ("unbounded")	
term	Element Declaration, Element Wildcard o ModelGroup	

Simple Type

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Simple Type Definition"
name	Cadena	Nombre local del tipo (vacío si es anónimo)
target namespace	Cadena	URI del espacio de nombres del tipo (vacío si es anónimo)
final	Secuencia de cadenas ("restriction"|"extension"|"list"|"union")	
context	Componente contenedor	
base type definition	Simple Type	
facets	Secuencia de facetas	
fundamental facets	Secuencia vacía (no implementada)	
variety	Cadena ("atomic"|"list"|"union")	
primitive type definition	Simple Type	
item type definition	(Sólo para tipos de lista) Simple Type	
member type definitions	(Sólo para tipos de unión) Secuencia de elementos Simple Type	

Type Alternative

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Type Alternative"
test	Registro de propiedades XPath	
type definition	Simple Type o Complex Type	

XPath Property Record

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
namespace bindings	Secuencia de funciones con propiedades ("prefix": string, "namespace": anyURI)	
default namespace	anyURI	
base URI	anyURI	El URI de base estático de la expresión XPath
expression	Cadena	La expresión XPath como cadena de texto

altova:schema(ComponentKind *as xs:string*, **Name** *as xs:QName*) **como (function(xs:string) como item()*)?** XP3.1 XQ3.1

Devuelve el tipo de componente que se indica en el primer argumento que tiene el mismo nombre que el que se indica en el segundo argumento. Para seguir navegando seleccione una de las propiedades del componente.

- Si esta propiedad es un componente seleccione una de sus propiedades para navegar hasta el siguiente nivel de profundidad. Puede repetir este paso para seguir navegando por el esquema.
- Si el componente es un valor atómico se devuelve este valor y no puede seguir navegando.

Nota: en las expresiones XQuery, el esquema se debe importar de forma explícita. En las expresiones XPath debe importar primero el esquema en el entorno de procesamiento, por ejemplo en XSLT con la instrucción **xslt:import**.

Ejemplos

- `import schema "" at "C:\Test\ExpReport.xsd";`
  `altova:schema("element declaration", xs:QName("OrgChart"))("type definition")`
  `("content type")("particles")[3]!.("term")("kind")`
  devuelve la propiedad `kind` del término del tercer componente `particles`. Este componente desciende de la declaración de elementos que tiene un `QName` de `OrgChart`

- `import schema "" at "C:\Test\ExpReport.xsd";`
  `let $typedef := altova:schema("type definition", xs:QName("emailType"))`
  `for $facet in $typedef ("facets")`
  `return [$facet ("kind"), $facet("value")]`
  devuelve, por cada **facet** de cada componente **emailType**, una matriz que contiene el tipo y el valor de ese elemento **facet**

Componentes y sus propiedades

- Assertion

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Assertion"
test	Registro de propiedades XPath	

- Attribute Declaration

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Attribute Declaration"
name	Cadena	Nombre local del atributo
target namespace	Cadena	URI del espacio de nombres del atributo
type definition	Simple Type o Complex Type	
scope	Una función con propiedades ("class":"Scope", "variety": "global" o "local", "parent": el Complex Type o Attribute Group contenedor)	
value constraint	Si está presente, una función con propiedades ("class": "Value Constraint", "variety": "fixed" o "default", "value": atomic value, "lexical form": string. Tenga en cuenta que la propiedad "value" no está disponible para los tipos namespace-sensitive	
inheritable	Booleano	

- Attribute Group Definition

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Attribute Group Definition"
name	Cadena	Nombre local del grupo de atributos
target namespace	Cadena	URI del espacio de nombres del grupo de atributos
attribute uses	Secuencia de (Attribute Use)	
attribute wildcard	Comodín de atributo opcional	

- Attribute Use

Nombre de la	Tipo de la propiedad	Valor de la propiedad

propiedad		
kind	Cadena	"Attribute Use"
required	Booleano	true si el atributo es obligatorio, false si es opcional
value constraint	Véase la declaración de atributos	
inheritable	Booleano	

Attribute Wildcard

Nombre de la propiedad	**Tipo de la propiedad**	**Valor de la propiedad**
kind	Cadena	"Wildcard"
namespace constraint	Función con propiedades ("class": "Namespace Constraint", "variety": "any"|"enumeration"|"not", "namespaces": secuencia de elementos xs:anyURI, "disallowed names": lista que contiene QNames y/o las cadenas "defined" y "definedSiblings"	
process contents	Cadena ("strict"|"lax"|"skip")	

Complex Type

Nombre de la propiedad	**Tipo de la propiedad**	**Valor de la propiedad**
kind	Cadena	"Complex Type"
name	Cadena	Nombre local del tipo (vacío si es anónimo)
target namespace	Cadena	URI del espacio de nombres del tipo (vacío si es anónimo)
base type definition	Definición del Complex Type	
final	Secuencia de cadenas ("restriction"|"extension")	
context	Secuencia vacía (not implemented)	
derivation method	Cadena ("restriction"|"extension")	
abstract	Booleano	
attribute uses	Secuencia de elementos Attribute Use	
attribute wildcard	Comodín de atributo opcional	
content type	Función con propiedades: ("class":"Content Type", "variety":string ("element-only"|"empty"|"mixed"|"simple"), particle: partícula opcional, "open content": función con propiedades	

	("class":"Open Content", "mode": string ("interleave"|"suffix"), "wildcard": Wildcard), "simple type definition": Simple Type)	
prohibited substitutions	Secuencia de cadenas ("restriction"|"extension")	
assertions	Secuencia de elementos Assertion	

Element Declaration

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Complex Type"
name	Cadena	Nombre local del tipo (vacío si es anónimo)
target namespace	Cadena	Namespace URI del tipo (vacío si es anónimo)
type definition	Simple Type o Complex Type	
type table	Función con propiedades ("class":"Type Table", "alternatives": secuencia de elementos Type Alternative, "default type definition": Simple Type o Complex Type)	
scope	Función con propiedades ("class":"Scope", "variety": ("global"|"local"), "parent": Complex Type opcional)	
value constraint	véase Attribute Declaration	
nillable	Booleano	
identity-constraint definitions	Secuencia de restricciones de identidad	
substitution group affiliations	Secuencia de declaraciones de elementos	
substitution group exclusions	Secuencia de cadenas ("restriction"|"extension")	
disallowed substitutions	Secuencia de cadenas ("restriction"|"extension"|"substitution")	
abstract	Booleano	

Element Wildcard

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Wildcard"

namespace constraint	Función con propiedades ("class": "Namespace Constraint", "variety": "any"|"enumeration"|"not", "namespaces": secuencia de xs:anyURI, "disallowed names": lista que contiene QNames y/o las cadenas "defined" y "definedSiblings"	
process contents	Cadena ("strict"|"lax"|"skip")	

Facet

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	El nombre de la faceta, por ejemplo "minLength" o "enumeration"
value	Depende de la faceta	El valor de la faceta
fixed	Booleano	
typed-value	Sólo para facetas de enumeración, Array(xs:anyAtomicType*)	Una matriz que contiene los valores de la enumeración, cada uno de los cuales puede ser una secuencia de valores atómicos. (Nota: para la faceta de enumeración, la propiedad "value" es un secuencia de cadenas, independientemente del tipo)

Identity Constraint

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Identity-Constraint Definition"
name	Cadena	Nombre local de la restricción
target namespace	Cadena	URI del espacio de nombres de la restricción
identity-constraint category	Cadena ("key"|"unique"|"keyRef")	
selector	Registro de propiedades XPath	
fields	Secuencia de registros de propiedades XPath	
referenced key	(Sólo para keyRef): Identity Constraint	La restricción clave correspondiente

Model Group

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Model Group"

compositor	Cadena ("sequence"|"choice"|"all")	
particles	Secuencia de partículas	

Model Group Definition

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Model Group Definition"
name	Cadena	Nombre local del grupo de modelos
target namespace	Cadena	URI del espacio de nombres del grupo de modelos
model group	Model Group	

Notation

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Notation Declaration"
name	Cadena	Nombre local de la notación
target namespace	Cadena	URI del espacio de nombres de la notación
system identifier	anyURI	
public identifier	Cadena	

Particle

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Particle"
min occurs	Número entero	
max occurs	Número entero o cadena ("unbounded")	
term	Element Declaration, Element Wildcard o ModelGroup	

Simple Type

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Simple Type Definition"
name	Cadena	Nombre local del tipo (vacío si es anónimo)
target namespace	Cadena	URI del espacio de nombres del tipo (vacío si es anónimo)

final	Secuencia de cadenas ("restriction"|"extension"|"list"|"union")	
context	Componente contenedor	
base type definition	Simple Type	
facets	Secuencia de facetas	
fundamental facets	Secuencia vacía (no implementada)	
variety	Cadena ("atomic"|"list"|"union")	
primitive type definition	Simple Type	
item type definition	(Sólo para tipos de lista) Simple Type	
member type definitions	(Sólo para tipos de unión) Secuencia de elementos Simple Type	

Type Alternative

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Type Alternative"
test	Registro de propiedades XPath	
type definition	Simple Type o Complex Type	

XPath Property Record

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
namespace bindings	Secuencia de funciones con propiedades ("prefix": string, "namespace": anyURI)	
default namespace	anyURI	
base URI	anyURI	El URI de base estático de la expresión XPath
expression	Cadena	La expresión XPath como cadena de texto

`altova:type(Node` *`as item?`*`) como (function(xs:string) como item()*)?` XP3.1 XQ3.1
La función **`altova:type`** indica un nodo de elemento o atributo de un documento XML y devuelve la información del tipo de nodo del PSVI (Conjunto de información posterior a la validación de esquemas).

Nota: el documento XML debe tener una declaración de esquema para que se pueda hacer referencia al esquema.

Ejemplos

- `for $element in //Email`

```
let $type := altova:type($element)
return $type
```
devuelve una función que contiene información sobre el tipo de nodo

- ```
  for $element in //Email
  let $type := altova:type($element)
  return $type ("kind")
  ```
  toma el componente de tipo del nodo (tipo simple o complejo) y devuelve el valor de la propiedad **kind** del componente

El parámetro **"_props"** devuelve las propiedades del componente seleccionado. Por ejemplo:

- ```
  for $element in //Email
  let $type := altova:type($element)
  return ($type ("kind"), $type ("_props"))
  ```
  toma el componente de tipo del nodo (tipo simple o complejo) y devuelve (i) el valor de la propiedad **kind** del componente y después (ii) las propiedades de ese componente

Componentes y sus propiedades

Assertion

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Assertion"
test	Registro de propiedades XPath	

Attribute Declaration

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Attribute Declaration"
name	Cadena	Nombre local del atributo
target namespace	Cadena	URI del espacio de nombres del atributo
type definition	Simple Type o Complex Type	
scope	Una función con propiedades ("class":"Scope", "variety": "global" o "local", "parent": el Complex Type o Attribute Group contenedor)	
value constraint	Si está presente, una función con propiedades ("class": "Value Constraint", "variety": "fixed" o "default", "value": atomic value, "lexical form": string. Tenga en cuenta que la propiedad "value" no está disponible para los tipos namespace-sensitive	

inheritable	Booleano	

Attribute Group Definition

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Attribute Group Definition"
name	Cadena	Nombre local del grupo de atributos
target namespace	Cadena	URI del espacio de nombres del grupo de atributos
attribute uses	Secuencia de (Attribute Use)	
attribute wildcard	Comodín de atributo opcional	

Attribute Use

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Attribute Use"
required	Booleano	true si el atributo es obligatorio, false si es opcional
value constraint	Véase la declaración de atributos	
inheritable	Booleano	

Attribute Wildcard

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Wildcard"
namespace constraint	Función con propiedades ("class": "Namespace Constraint", "variety": "any"|"enumeration"|"not", "namespaces": secuencia de elementos xs:anyURI, "disallowed names": lista que contiene QNames y/o las cadenas "defined" y "definedSiblings"	
process contents	Cadena ("strict"|"lax"|"skip")	

Complex Type

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Complex Type"
name	Cadena	Nombre local del tipo (vacío si es anónimo)

target namespace	Cadena	URI del espacio de nombres del tipo (vacío si es anónimo)
base type definition	Definición del Complex Type	
final	Secuencia de cadenas ("restriction"|"extension")	
context	Secuencia vacía (not implemented)	
derivation method	Cadena ("restriction"|"extension")	
abstract	Booleano	
attribute uses	Secuencia de elementos Attribute Use	
attribute wildcard	Comodín de atributo opcional	
content type	Función con propiedades: ("class":"Content Type", "variety":string ("element-only"|"empty"|"mixed"|"simple"), particle: partícula opcional, "open content": función con propiedades ("class":"Open Content", "mode": string ("interleave"|"suffix"), "wildcard": Wildcard), "simple type definition": Simple Type)	
prohibited substitutions	Secuencia de cadenas ("restriction"|"extension")	
assertions	Secuencia de elementos Assertion	

Element Declaration

Nombre de la propiedad	**Tipo de la propiedad**	**Valor de la propiedad**
kind	Cadena	"Complex Type"
name	Cadena	Nombre local del tipo (vacío si es anónimo)
target namespace	Cadena	Namespace URI del tipo (vacío si es anónimo)
type definition	Simple Type o Complex Type	
type table	Función con propiedades ("class":"Type Table", "alternatives": secuencia de elementos Type Alternative, "default type definition": Simple Type o Complex Type)	
scope	Función con propiedades ("class":"Scope", "variety": ("global"|"local"), "parent": Complex Type opcional)	
value constraint	véase Attribute Declaration	
nillable	Booleano	

identity-constraint definitions	Secuencia de restricciones de identidad	
substitution group affiliations	Secuencia de declaraciones de elementos	
substitution group exclusions	Secuencia de cadenas ("restriction"|"extension")	
disallowed substitutions	Secuencia de cadenas ("restriction"|"extension"|"substitution")	
abstract	Booleano	

Element Wildcard

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Wildcard"
namespace constraint	Función con propiedades ("class": "Namespace Constraint", "variety": "any"|"enumeration"|"not", "namespaces": secuencia de xs:anyURI, "disallowed names": lista que contiene QNames y/o las cadenas "defined" y "definedSiblings"	
process contents	Cadena ("strict"|"lax"|"skip")	

Facet

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	El nombre de la faceta, por ejemplo "minLength" o "enumeration"
value	Depende de la faceta	El valor de la faceta
fixed	Booleano	
typed-value	Sólo para facetas de enumeración, Array(xs:anyAtomicType*)	Una matriz que contiene los valores de la enumeración, cada uno de los cuales puede ser una secuencia de valores atómicos. (Nota: para la faceta de enumeración, la propiedad "value" es un secuencia de cadenas, independientemente del tipo)

Identity Constraint

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Identity-Constraint Definition"

name	Cadena	Nombre local de la restricción
target namespace	Cadena	URI del espacio de nombres de la restricción
identity-constraint category	Cadena ("key"|"unique"|"keyRef")	
selector	Registro de propiedades XPath	
fields	Secuencia de registros de propiedades XPath	
referenced key	(Sólo para keyRef): Identity Constraint	La restricción clave correspondiente

Model Group

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Model Group"
compositor	Cadena ("sequence"|"choice"|"all")	
particles	Secuencia de partículas	

Model Group Definition

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Model Group Definition"
name	Cadena	Nombre local del grupo de modelos
target namespace	Cadena	URI del espacio de nombres del grupo de modelos
model group	Model Group	

Notation

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Notation Declaration"
name	Cadena	Nombre local de la notación
target namespace	Cadena	URI del espacio de nombres de la notación
system identifier	anyURI	
public identifier	Cadena	

Particle

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad

kind	Cadena	"Particle"
min occurs	Número entero	
max occurs	Número entero o cadena ("unbounded")	
term	Element Declaration, Element Wildcard o ModelGroup	

Simple Type

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Simple Type Definition"
name	Cadena	Nombre local del tipo (vacío si es anónimo)
target namespace	Cadena	URI del espacio de nombres del tipo (vacío si es anónimo)
final	Secuencia de cadenas ("restriction"|"extension"|"list"|"union")	
context	Componente contenedor	
base type definition	Simple Type	
facets	Secuencia de facetas	
fundamental facets	Secuencia vacía (no implementada)	
variety	Cadena ("atomic"|"list"|"union")	
primitive type definition	Simple Type	
item type definition	(Sólo para tipos de lista) Simple Type	
member type definitions	(Sólo para tipos de unión) Secuencia de elementos Simple Type	

Type Alternative

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
kind	Cadena	"Type Alternative"
test	Registro de propiedades XPath	
type definition	Simple Type o Complex Type	

XPath Property Record

Nombre de la propiedad	Tipo de la propiedad	Valor de la propiedad
namespace bindings	Secuencia de funciones con propiedades ("prefix": string, "namespace": anyURI)	

default namespace	anyURI	
base URI	anyURI	El URI de base estático de la expresión XPath
expression	Cadena	La expresión XPath como cadena de texto

12.2.2.1.7 Funciones XPath/XQuery: Secuencia

Las funciones de extensión de Altova para trabajar con secuencias pueden utilizarse en expresiones XPath y XQuery y ofrecen funciones adicionales para el procesamiento de datos. Estas funciones se pueden usar con los motores **XPath 3.0** y **XQuery 3.0** de Altova. Están disponibles en contextos XPath/XQuery.

Nota sobre el nombre de las funciones y lenguajes

Puede utilizar todas las funciones de extensión de Altova en sus expresiones XPath/XQuery. Con ellas conseguirá funciones adicionales no disponibles en la biblioteca de funciones estándar de XPath, XQuery y XSLT. Las funciones de extensión de Altova están en el **espacio de nombres** `http://www.altova.com/xslt-extensions` y en esta sección se presentan con el prefijo, que se supone estará enlazado al espacio de nombres señalado. Tenga en cuenta que en futuras versiones del producto algunas funciones pueden dejar de ser compatibles o su comportamiento puede cambiar. Por tanto, consulte siempre la documentación del producto para conocer el funcionamiento de estas funciones en cada versión del producto.

Funciones XPath (en expresiones XPath en XSLT):	XP1 XP2 XP3.1.1
Funciones XSLT (en expresiones XPath en XSLT):	XSLT1 XSLT2 XSLT3
Funciones XQuery (en expresiones XQuery en XQuery):	XQ1 XQ3.1

▼ attributes [altova:]

altova:attributes(NombreAtributo *as xs:string***)** como **attribute()*** XP3.1 XQ3.1
Devuelve todos los atributos cuyo nombre local coincida con el nombre dado como argumento de entrada (`NombreAtributo`). La búsqueda tiene en cuenta el uso de mayúsculas y minúsculas y se lleva a cabo en el eje `attribute::`.

▣ *Ejemplos*

- **altova:attributes**("MiAtributo") devuelve MiAtributo()*

altova:attributes(NombreAtributo *as xs:string*, **OpcionesBúsqueda** *as xs:string***)** como **attribute()*** XP3.1 XQ3.1
Devuelve todos los atributos cuyo nombre local coincida con el nombre dado como argumento de entrada

(NombreAtributo). La búsqueda tiene en cuenta el uso de mayúsculas y minúsculas y se lleva a cabo en el eje attribute::. El segundo argumento es una cadena con marcas de búsqueda. Estas son las marcas disponibles:

r = habilita la búsqueda de expresiones regulares. En este caso, NombreAtributo debe ser una cadena de búsqueda de expresión regular;
i = la búsqueda no tiene en cuenta el uso de mayúsculas y minúsculas;
p = incluye el prefijo de espacio de nombres en la búsqueda. En este caso, NombreAtributo debe contener el prefijo de espacio de nombres (p. ej.: MiAtributo).

Las marcas pueden escribirse en cualquier orden y no hace falta utilizar todas. Si usa marcas no válidas, se genera un error. También puede usar una cadena vacía para el segundo argumento. Esto tiene el mismo efecto que usar solo el primer argumento. Sin embargo, no está permitido usar una secuencia vacía.

Ejemplos

- **altova:attributes**("MiAtributo", "rip") devuelve MiAtributo()*
- **altova:attributes**("MiAtributo", "pri") devuelve MiAtributo()*
- **altova:attributes**("MiAtributo", "") devuelve MiAtributo()*
- **altova:attributes**("MiAtributo", "Rip") devuelve un error de marca desconocida.
- **altova:attributes**("MiAtributo",) devuelve un error diciendo que falta el segundo argumento.

elements [altova:]

altova:elements(NombreElemento *as xs:string***)** como **elemento()*** XP3.1 XQ3.1
Devuelve todos los elementos cuyo nombre local coincida con el nombre dado como argumento de entrada (NombreElemento). La búsqueda tiene en cuenta el uso de mayúsculas y minúsculas y se lleva a cabo en el eje child::.

Ejemplos

- **altova:elements**("MiElemento") devuelve MiElemento()*

altova:elements(NombreElemento *as xs:string*, **OpcionesBúsqueda** *as xs:string***)** como **elemento()*** XP3.1 XQ3.1
Devuelve todos los elementos cuyo nombre local coincida con el nombre dado como argumento de entrada (NombreElemento). La búsqueda tiene en cuenta el uso de mayúsculas y minúsculas y se lleva a cabo en el eje child::. El segundo argumento es una cadena con marcas de búsqueda. Estas son las marcas disponibles:

r = habilita la búsqueda de expresiones regulares. En este caso, NombreElemento debe ser una cadena de búsqueda de expresión regular;
i = la búsqueda no tiene en cuenta el uso de mayúsculas y minúsculas;
p = incluye el prefijo de espacio de nombres en la búsqueda. En este caso, NombreElemento debe contener el prefijo de espacio de nombres (p. ej.: MiElemento).

Las marcas pueden escribirse en cualquier orden y no hace falta utilizar todas. Si usa marcas no válidas, se genera un error. También puede usar una cadena vacía para el segundo argumento. Esto tiene el mismo efecto que usar solo el primer argumento. Sin embargo, no está permitido usar una secuencia

vacía.

Ejemplos

- **altova:elements**("MiElemento", "rip") devuelve MiElemento()*
- **altova:elements**("MiElemento", "pri") devuelve MiElemento()*
- **altova:elements**("MiElemento", "") devuelve MiElemento()*
- **altova:elements**("MiElemento", "Rip") devuelve un error de marca desconocida.
- **altova:elements**("MiElemento",) devuelve un error diciendo que falta el segundo argumento.

find-first [altova:]

altova:find-first((Secuencia *()**)*, **(Condición(Elemento-Secuencia** *como xs:boolean*))
como **item()?** XP3.1 XQ3.1
Esta función toma dos argumentos. El primero es una secuencia de uno o varios elementos de cualquier tipo de datos. El segundo argumento, **Condición**, es una referencia a una función XPath que toma un argumento (es decir, su aridad es 1) y devuelve un valor binario. Cada elemento de **Secuencia** se envía a su vez a la función a la que se hace referencia en **Condición**. Nota: recuerde que esta función solo toma un argumento. El primer elemento de **Secuencia** que consiga que la función de **Condición** dé **true()** como resultado se devuelve como resultado de **find-first** y la iteración se detiene.

Ejemplos

- **altova:find-first**(5 to 10, function($a) {$a mod 2 = 0}) devuelve xs:integer 6
  El argumento **Condición** remite a la función inline XPath 3.0 **function()**, que declara una función inline llamada **$a** y después la define. Cada elemento del argumento Secuencia de **find-first** se envía a su vez como valor de entrada a **$a**. El valor de entrada se prueba en la condición en la definición de función ($a mod 2 = 0). El primer valor de entrada que cumpla la condición se devuelve como resultado de **find-first** (en este caso **6**).

- **altova:find-first**((1 to 10), (function($a) {$a+3=7})) devuelve xs:integer 4

Más ejemplos
Si existe el archivo **C:\Temp\Customers.xml**:

- **altova:find-first**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) devuelve xs:string C:\Temp\Customers.xml

Si no existe el archivo **C:\Temp\Customers.xml** pero existe **http://www.altova.com/index.html**:

- **altova:find-first**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) devuelve xs:string http://www.altova.com/index.html

Si no existe el archivo **C:\Temp\Customers.xml** y tampoco existe **http://www.altova.com/index.html**:

- **altova:find-first**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) no devuelve ningún resultado

Notas sobre los ejemplos anteriores

- La función XPath 3.0 `doc-available` toma un solo argumento de cadena, que se usa como URI, y devuelve `true` si en el URI dado se encuentra un nodo de documento. El documento que está en el URI dado debe ser un documento XML.
- La función `doc-available` se puede usar para **Condición**, el segundo argumento de `find-first`, porque solamente toma un argumento (aridad=1), porque toma un `item()` como entrada (una cadena que se usa como URI) y devuelve un valor binario.
- Recuerde que solamente se hace referencia a la función `doc-available` pero no se le llama. El sufijo `#1` que se anexa a la función indica una función cuya aridad es 1. Es decir, `doc-available#1` simplemente significa "*Utilizar la función doc-availabe() que tiene aridad=1, pasándole como solo argumento a su vez cada uno de los elementos de la primera secuencia.*" Como resultado, se pasarán las dos cadenas a **doc-available()**, que utiliza la cadena como URI y prueba si existe un nodo de documento en el URI. Si existe, entonces `doc-available()` da como resultado `true()` y esa cadena se devuelve como resultado de la función **find-first**. Nota sobre la función `doc-available()`: las rutas de acceso relativas se resuelven en relación al URI base actual, que es por defecto el URI del documento XML desde el que se carga la función.

▼ find-first-combination [altova:]

altova:find-first-combination((Sec-01 *como item()**, **(Sec-02** *como item()**, **(Condición(Elem-Sec-01, Elem-Sec-02** *como xs:boolean*)) como **item()*** XP3.1 XQ3.1

Esta función toma tres argumentos:

- Los dos primeros (**Sec-01** y **Sec-02**) son secuencias de uno o más elementos de cualquier tipo de datos.
- El tercero (**Condición**) es una referencia a una función XPath que toma dos argumentos (su aridad es 2) y devuelve un valor binario.

Los elementos de **Sec-01** y **Sec-02** se pasan en pares ordenados (cada par está formado por un elemento de cada secuencia) como argumentos de la función de **Condición**. Los pares se ordenan de la siguiente manera:

```
Si   Sec-01 = X1, X2, X3 ... Xn
Y  Sec-02 = Y1, Y2, Y3 ... Yn
Entonces (X1 Y1), (X1 Y2), (X1 Y3) ... (X1 Yn), (X2 Y1), (X2 Y2) ... (Xn Yn)
```

El primer par ordenado que consiga que la función de **Condición** dé como resultado **true()** se devuelve como resultado de **find-first-combination**. Recuerde que (i) si la función de **Condición** recorre los pares de argumentos dados y no consigue dar **true()** como resultado ni una vez, entonces **find-first-combination** devuelve *Sin resultados;* (ii) el resultado de **find-first-combination** siempre será un par de elementos (de cualquier tipo de datos) o ningún elemento.

▬ *Ejemplos*

- **altova:find-first-pair**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 32}) devuelve la secuencia de `xs:integers (11, 21)`
- **altova:find-first-pair**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 33}) devuelve la secuencia de `xs:integers (11, 22)`
- **altova:find-first-pair**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 34}) devuelve

la secuencia de `xs:integers (11, 23)`

▼ find-first-pair [altova:]

`altova:find-first-pair((Sec-01` *como item()`), (Sec-02` *como item()**`), (Condición( Elem-Sec-01, Elem-Sec-02` *como xs:boolean*`))`** como **`item()*`** XP3.1 XQ3.1
Esta función toma tres argumentos:

- Los dos primeros (**`Sec-01`** y **`Sec-02`**) son secuencias de uno o más elementos de cualquier tipo de datos.
- El tercero (**`Condición`**) es una referencia a una función XPath que toma dos argumentos (su aridad es 2) y devuelve un valor binario.

Los elementos de **`Sec-01`** y **`Sec-02`** se pasan en pares ordenados como argumentos de la función de **`Condición`**. Los pares se ordenan de la siguiente manera:

```
Si   Sec-01 = X1, X2, X3 ... Xn
Y  Sec-02 = Y1, Y2, Y3 ... Yn
Entonces (X1 Y1), (X2 Y2), (X3 Y3) ... (Xn Yn)
```

El primer par ordenado que consiga que la función de **`Condición`** dé como resultado **`true()`** se devuelve como resultado de **`find-first-pair`**. Recuerde que (i) si la función de **`Condición`** recorre los pares de argumentos dados y no consigue dar **`true()`** como resultado ni una vez, entonces **`find-first-pair`** devuelve *Sin resultados;* (ii) el resultado de **`find-first-pair`** siempre será un par de elementos (de cualquier tipo de datos) o ningún elemento.

▼ *Ejemplos*

- **`altova:find-first-pair`**`(11 to 20, 21 to 30, function($a, $b) {$a+$b = 32})` devuelve la secuencia de `xs:integers (11, 21)`
- **`altova:find-first-pair`**`(11 to 20, 21 to 30, function($a, $b) {$a+$b = 33})` devuelve *Sin resultados*

Observe que en los dos ejemplos anteriores el orden de los pares es: `(11, 21) (12, 22) (13, 23)...(20, 30)`. Por ese motivo el segundo ejemplo no obtiene resultados (porque ningún par ordenado consigue sumar 33).

▼ find-first-pair-pos [altova:]

`altova:find-first-pair-pos((Sec-01` *como item()`), (Sec-02` *como item()**`), (Condición( Elem-Sec-01, Elem-Sec-02` *como xs:boolean*`))`** como **`xs:integer`** XP3.1 XQ3.1
Esta función toma tres argumentos:

- Los dos primeros (**`Sec-01`** y **`Sec-02`**) son secuencias de uno o más elementos de cualquier tipo de datos.
- El tercero (**`Condición`**) es una referencia a una función XPath que toma dos argumentos (su aridad es 2) y devuelve un valor binario.

Los elementos de **`Sec-01`** y **`Sec-02`** se pasan en pares ordenados como argumentos de la función de **`Condición`**. Los pares se ordenan de la siguiente manera:

```
Si   Sec-01 = X1, X2, X3 ... Xn
Y  Sec-02 = Y1, Y2, Y3 ... Yn
Entonces (X1 Y1), (X2 Y2), (X3 Y3) ... (Xn Yn)
```

La posición de índice del primer par ordenado que consiga que la función de **Condición** dé como resultado **true()** se devuelve como resultado de **find-first-pair-pos**. Recuerde que si la función de **Condición** recorre los pares de argumentos dados y no da como resultado **true()** ni una sola vez, entonces **find-first-pair-pos** devuelve *Sin resultados*.

Ejemplos

- **altova:find-first-pair**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 32}) devuelve 1
- **altova:find-first-pair**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 33}) devuelve *Sin resultados*

Observe que en los dos ejemplos anteriores el orden de los pares es: (11, 21) (12, 22) (13, 23)...(20, 30). En el primer ejemplo el primer par consigue que la función de **Condición** dé como resultado **true()** y, por tanto, se devuelve la posición de índice que tienen en la secuencia (**1**). El segundo ejemplo, sin embargo, devuelve *Sin resultados* porque ningún par consigue sumar 33.

find-first-pos [altova:]

altova:find-first-pos((Secuencia *como item()**)**, **(Condición(Elem-Sec** *como xs:boolean***))** como **xs:integer** XP3.1 XQ3.1

Esta función toma dos argumentos. El primer argumento es una secuencia de uno o varios elementos de cualquier tipo. El segundo argumento (**Condición**) es una referencia a una función XPath que toma un argumento (su aridad es 1) y devuelve un valor binario. Cada elemento de **Secuencia** se envía a su vez a la función a la que se hace referencia en **Condición**. (Recuerde que esta función toma un solo argumento.) El primer elemento de **Secuencia** que consiga que la función de **Condición** dé como resultado **true()** devuelve la posición de índice que tiene en **Secuencia** como resultado de **find-first-pos** y la iteración se detiene.

Ejemplos

- **altova:find-first-pos**(5 to 10, function($a) {$a mod 2 = 0}) devuelve xs:integer 2

El argumento **Condición** hace referencia a la función inline XPath 3.0 **function()**, que declara una función inline llamada **$a** y después la define. Cada elemento del argumento **Sequence** de **find-first-pos** se pasa a su vez como valor de entrada de **$a**. El valor de entrada se prueba en la condición de la definición de función ($a mod 2 = 0). La posición de índice que tiene en la secuencia el primer valor de entrada que cumple la condición se devuelve como resultado de **find-first-pos** (en este caso es la posición de índice **2**, porque **6** es el primer valor (de la secuencia) que cumple la condición y su posición de índice en la secuencia es **2**).

- **altova:find-first-pos**((2 to 10), (function($a) {$a+3=7})) devuelve xs:integer 3

Más ejemplos

Si existe el archivo **C:\Temp\Customers.xml**:

- **altova:find-first-pos**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) devuelve 1

Si no existe el archivo **C:\Temp\Customers.xml** pero existe **http://www.altova.com/index.html**:

- **altova:find-first-pos**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) devuelve 2

Si no existe el archivo **C:\Temp\Customers.xml** y tampoco existe **http://www.altova.com/index.html**:

- **altova:find-first-pos**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) no devuelve ningún resultado

Notas sobre los ejemplos anteriores

- La función XPath 3.0 doc-available toma un solo argumento de cadena, que se usa como URI, y devuelve true si en el URI dado se encuentra un nodo de documento. El documento que está en el URI dado debe ser un documento XML.
- La función doc-available se puede usar para **Condición**, el segundo argumento de find-first-pos, porque solamente toma un argumento (aridad=1), porque toma un item() como entrada (una cadena que se usa como URI) y devuelve un valor binario.
- Recuerde que solamente se hace referencia a la función doc-available pero no se le llama. El sufijo #1 que se anexa a la función indica una función cuya aridad es 1. Es decir, doc-available#1 simplemente significa "*Utilizar la función doc-availabe() que tiene aridad=1, pasándole como solo argumento a su vez cada uno de los elementos de la primera secuencia.*" Como resultado, se pasarán las dos cadenas a **doc-available()**, que utiliza la cadena como URI y prueba si existe un nodo de documento en el URI. Si existe, entonces doc-available() da como resultado true() y esa cadena se devuelve como resultado de la función **find-first-pos**. Nota sobre la función doc-available(): las rutas de acceso relativas se resuelven en relación al URI base actual, que es por defecto el URI del documento XML desde el que se carga la función.

for-each-attribute-pair [altova:]

altova:for-each-attribute-pair(Seq1 *como element()?*, **Seq2** *como element()?*, **Function** *como function()*) como **item()*** XP3.1 XQ3.1

Los primeros dos argumentos identifican dos elementos cuyos atributos se usan para construir pares de atributos donde uno de los atributos del par se obtiene del primer elemento y el otro atributo del segundo elemento. Los pares de atributos se seleccionan basándose en que tienen el mismo nombre y se ordenan alfabéticamente por grupos. Si un atributo no tiene un atributo correspondiente en el otro elemento, entonces el par está "desarticulado", lo que significa que tiene un solo miembro. El elemento de la función (tercer argumento Function) se aplica por separado a cada par de la secuencia de pares (articulados y desarticulados) y el resultado es una secuencia de elementos.

Ejemplos

- **altova:for-each-attribute-pair**(/Example/Test-A, /Example/Test-B, function($a, $b) {$a+b}) devuelve...

(2, 4, 6) si
<Test-A att1="1" att2="2" att3="3" />
<Test-B att1="1" att2="2" att3="3" />

(2, 4, 6) si
<Test-A att2="2" att1="1" att3="3" />
<Test-B att3="3" att2="2" att1="1" />

(2, 6) si
<Test-A att4="4" att1="1" att3="3" />
<Test-B att3="3" att2="2" att1="1" />

Nota: El resultado **(2, 6)** *se obtiene mediante la siguiente acción:* **(1+1, ()+2, 3+3, 4+())**. *Si uno de los operandos es la secuencia vacía, como en el caso de los elementos 2 y 4, entonces el resultado de la suma es una secuencia vacía.*

- **altova:for-each-attribute-pair**(/Example/Test-A, /Example/Test-B, concat#2) devuelve...

(11, 22, 33) si
<Test-A att1="1" att2="2" att3="3" />
<Test-B att1="1" att2="2" att3="3" />

(11, 2, 33, 4) si
<Test-A att4="4" att1="1" att3="3" />
<Test-B att3="3" att2="2" att1="1" />

▼ for-each-combination [altova:]

altova:for-each-combination(PrimeraSecuencia *como item()**, **SegundaSecuencia** *como item()** *, **función($i,$j){$i || $j})** como **item()*** XP3.1 XQ3.1
Los elementos de las dos secuencias en los primeros dos argumentos se combinan de forma que el primer elemento de la primera secuencia se combina, en orden, una vez con cada elemento de la segunda secuencia. La función dada como tercer argumento se aplica a cada una de las combinaciones de la secuencia resultante y da como resultado una secuencia de elementos (*véase ejemplo*).

▣ *Ejemplos*

- **altova:for-each-combination**(('a', 'b', 'c'), ('1', '2', '3'), function($i, $j) {$i || $j}) devuelve ('a1', 'a2', 'a3', 'b1', 'b2', 'b3', 'c1', 'c2', 'c3')

▼ for-each-matching-attribute-pair [altova:]

altova:for-each-matching-attribute-pair(Seq1 *como element()?*, **Seq2** *como element()?*, **Function** *como function()***)** como **item()*** XP3.1 XQ3.1
Los primeros dos argumentos identifican dos elementos cuyos atributos se usan para construir pares de atributos donde un atributo de cada par se obtiene del primer elemento y el otro atributo del par se obtiene del segundo elemento. Los pares de elementos se seleccionan basándose en que tienen el mismo nombre y se ordenan alfabéticamente por grupos. Si un atributo no tiene un atributo correspondiente en el otro elemento, entonces no se construye ningún par. El elemento de la función (tercer argumento

`Function`) se aplica por separado a cada par de la secuencia de pares (articulados y desarticulados) y el resultado es una secuencia de elementos.

Ejemplos

- **`altova:for-each-matching-attribute-pair`**`(/Example/Test-A, /Example/Test-B, function($a, $b){$a+b})` devuelve...

```
(2, 4, 6) if
<Test-A att1="1" att2="2" att3="3" />
<Test-B att1="1" att2="2" att3="3" />

(2, 4, 6) if
<Test-A att2="2" att1="1" att3="3" />
<Test-B att3="3" att2="2" att1="1" />

(2, 6) if
<Test-A att4="4" att1="1" att3="3" />
<Test-B att3="3" att2="2" att3="1" />
```

- **`altova:for-each-matching-attribute-pair`**`(/Example/Test-A, /Example/Test-B, concat#2)` devuelve...

```
(11, 22, 33) if
<Test-A att1="1" att2="2" att3="3" />
<Test-B att1="1" att2="2" att3="3" />

(11, 33) if
<Test-A att4="4" att1="1" att3="3" />
<Test-B att3="3" att2="2" att1="1" />
```

▼ substitute-empty [altova:]

`altova:substitute-empty(PrimeraSecuencia` *`as item()*`**`, SegundaSecuencia`** *`as item()`***`)`** como **`item()*`** XP3.1 XQ3.1

Si **`PrimeraSecuencia`** está vacío, la función devuelve **`SegundaSecuencia`**. Si **`PrimeraSecuencia`** no está vacío, la función devuelve **`PrimeraSecuencia`**.

Ejemplos

- **`altova:substitute-empty`**`( (1,2,3), (4,5,6) )` devuelve `(1,2,3)`
- **`altova:substitute-empty`**`( (), (4,5,6) )` devuelve `(4,5,6)`

12.2.2.1.8 Funciones XPath/XQuery: Cadena

Las funciones de extensión de Altova para trabajar con cadenas pueden utilizarse en expresiones XPath y XQuery y ofrecen funciones adicionales para el procesamiento de datos. Estas funciones se pueden usar con los motores **XPath 3.0** y **XQuery 3.0** de Altova. Están disponibles en contextos XPath/XQuery.

Nota sobre el nombre de las funciones y lenguajes

Puede utilizar todas las funciones de extensión de Altova en sus expresiones XPath/XQuery. Con ellas conseguirá funciones adicionales no disponibles en la biblioteca de funciones estándar de XPath, XQuery y XSLT. Las funciones de extensión de Altova están en el **espacio de nombres** **`http://www.altova.com/xslt-extensions`** y en esta sección se presentan con el prefijo, que se supone estará enlazado al espacio de nombres señalado. Tenga en cuenta que en futuras versiones del producto algunas funciones pueden dejar de ser compatibles o su comportamiento puede cambiar. Por tanto, consulte siempre la documentación del producto para conocer el funcionamiento de estas funciones en cada versión del producto.

Funciones XPath (en expresiones XPath en XSLT):	XP1 XP2 XP3.1.1
Funciones XSLT (en expresiones XPath en XSLT):	XSLT1 XSLT2 XSLT3
Funciones XQuery (en expresiones XQuery en XQuery):	XQ1 XQ3.1

▼ camel-case [altova:]

`altova:camel-case(CadenaEntrada` *como xs:string*`)` como `xs:string` XP3.1 XQ3.1
Devuelve la cadena de entrada **`CadenaEntrada`** escrita en CamelCase. La cadena se analiza usando la expresión regular **`'\s'`** (que es la forma abreviada del carácter espacio en blanco). El primer carácter que no sea un espacio en blanco situado después de un espacio en blanco o de una secuencia de espacios en blanco consecutivos se pondrá en mayúsculas. El primer carácter de la cadena de salida se pondrá en mayúsculas.

Ejemplos

- **`altova:camel-case`**`("max")` devuelve `Max`
- **`altova:camel-case`**`("max max")` devuelve `Max Max`
- **`altova:camel-case`**`("file01.xml")` devuelve `File01.xml`
- **`altova:camel-case`**`("file01.xml file02.xml")` devuelve `File01.xml File02.xml`
- **`altova:camel-case`**`("file01.xml  file02.xml")` devuelve `File01.xml  File02.xml`
- **`altova:camel-case`**`("file01.xml -file02.xml")` devuelve `File01.xml -file02.xml`

`altova:camel-case(CadenaEntrada` *como xs:string*`,` **`CaracteresDivisión`** *como xs:string*`,` **`EsExpReg`** *como xs:boolean*`)` como `xs:string` XP3.1 XQ3.1
Devuelve la cadena de entrada **`CadenaEntrada`** escrita en CamelCase usando los **`CaracteresDivisón`** para determinar qué caracteres desencadenan el siguiente uso de mayúsculas. El argumento **`CaracteresDivisón`** se usa como expresión regular **`cuando EsExpReg = true()`** o como caracteres planos cuando r **`EsExpReg = false()`**. El primer carácter de la cadensa de salida se escribe con mayúsculas.

Ejemplos

- **`altova:camel-case`**`("setname getname", "set|get", true())` devuelve `setName getName`
- **`altova:camel-case`**`("altova\documents\testcases", "\", false())` devuelve `Altova\Documents\Testcases`

▼ char [altova:]

altova:char(Posición *as xs:integer*) como **xs:string** XP3.1 XQ3.1
Devuelve una cadena que contiene el carácter que está en la posición indicada por el argumento **Posición** en la cadena que se obtiene al convertir el valor del elemento de contexto en `xs:string`. La cadena resultante estará vacía si en la posición indicada no existe ningún carácter.

⊟ *Ejemplos*

Si el elemento de contexto es **1234ABCD**:

- **altova:char**(2) devuelve 2
- **altova:char**(5) devuelve A
- **altova:char**(9) devuelve la cadena vacía
- **altova:char**(-2) devuelve la cadena vacía

altova:char(CadenaEntrada *as xs:string*, **Posición** *as xs:integer*) como **xs:string** XP3.1 XQ3.1
Devuelve una cadena que contiene el carácter que está en la posición indicada por el argumento **Posición** en la cadena dada por el argumento **CadenaEntrada**. La cadena resultante estará vacía si en la posición indicada no existe ningún carácter.

⊟ *Ejemplos*

- **altova:char**("2014-01-15", 5) devuelve -
- **altova:char**("USA", 1) devuelve U
- **altova:char**("USA", 1) devuelve la cadena vacía
- **altova:char**("USA", -2) devuelve la cadena vacía

▼ create-hash-from-string [altova:]

altova:create-hash-from-string(InputString *como xs:string*) como **xs:string** XP2 XQ1 XP3.1 XQ3.1
altova:create-hash-from-string(InputString *como xs:string*, **HashAlgo** *as xs:string*) como **xs:string** XP2 XQ1 XP3.1 XQ3.1
Genera una cadena hash a partir de InputString usando el algoritmo de hash especificado por el argumento `HashAlgo`. Se pueden usar los siguientes algoritmos de hash (en mayúsculas o minúsculas): **MD5**, **SHA-1**, **SHA-224**, **SHA-256**, **SHA-384**, **SHA-512**. Si no se especifica el segundo argumento (*véase la primera instrucción*) se usa el algoritmo de hash **SHA-256**.

⊞ *Ejemplos*

- **altova:create-hash-from-string**('abc') devuelve una cadena hash generada usando el algoritmo de hash **SHA-256**.
- **altova:create-hash-from-string**('abc', 'md5') devuelve una cadena hash generada usando el algoritmo de hash **MD5**.
- **altova:create-hash-from-string**('abc', 'MD5') devuelve una cadena hash generada usando el algoritmo de hash **MD5**.

▼ first-chars [altova:]

altova:first-chars(X *as xs:integer*) como **xs:string** XP3.1 XQ3.1
Devuelve una cadena que contiene los **X** primeros caracteres de la cadena que se obtiene al convertir el valor del elemento de contexto en `xs:string`.

⊟ *Ejemplos*

Si el elemento de contexto es **1234ABCD**:

- `altova:first-chars(2)` devuelve `12`
- `altova:first-chars(5)` devuelve `1234A`
- `altova:first-chars(9)` devuelve `1234ABCD`

altova:first-chars(CadenaEntrada *as xs:string*, **X** *as xs:integer*) como **xs:string** XP3.1 XQ3.1
Devuelve una cadena que contiene los **X** primeros caracteres de la cadena dada como argumento **CadenaEntrada**.

⊟ *Ejemplos*

- `altova:first-chars("2014-01-15", 5)` devuelve `2014-`
- `altova:first-chars("USA", 1)` devuelve `U`

▼ format-string [altova:]

altova:format-string(InputString *como xs:string*, **FormatSequence** *como item()**) como **xs:string** XP3.1 XQ3.1
La cadena de entrada (primer argumento) contiene parámetros posicionales (`%1`, `%2`, etc). Cada parámetro es reemplazado por el elemento cadena ubicado en la posición correspondiente de la secuencia de formato (enviada como segundo argumento). Por tanto, el primer elemento de la secuencia de formato reemplaza al parámetro posicional `%1`, el segundo elemento reemplaza a `%2` y así sucesivamente. La función devuelve esta secuencia con formato que contiene los elementos de reemplazo. Si no existe una cadena para alguno de los parámetros posicionales, entonces se devuelve ese mismo parámetro posicional. Esto ocurre cuando el índice de un parámetro posicional es mayor que el número de elementos de la secuencia de formato.

⊞ *Ejemplos*

- `altova:format-string('Hello %1, %2, %3', ('Jane','John','Joe'))` devuelve `"Hello Jane, John, Joe"`
- `altova:format-string('Hello %1, %2, %3', ('Jane','John','Joe', 'Tom'))` devuelve `"Hello Jane, John, Joe"`
- `altova:format-string('Hello %1, %2, %4', ('Jane','John','Joe', 'Tom'))` devuelve `"Hello Jane, John, Tom"`
- `altova:format-string('Hello %1, %2, %4', ('Jane','John','Joe'))` devuelve `"Hello Jane, John, %4"`

▼ last-chars [altova:]

altova:last-chars(X *as xs:integer*) como **xs:string** XP3.1 XQ3.1
Devuelve una cadena que contiene los X últimos caracteres de la cadena que se obtiene al convertir el valor del elemento de contexto en `xs:string`.

Ejemplos

Si el elemento de contexto es **1234ABCD**:

- **altova:last-chars**(2) devuelve CD
- **altova:last-chars**(5) devuelve 4ABCD
- **altova:last-chars**(9) devuelve 1234ABCD

altova:last-chars(CadenaEntrada *as xs:string*, **X** *as xs:integer***)** como **xs:string** XP3.1 XQ3.1
Devuelve una cadena que contiene los **X** últimos caracteres de la cadena dada como argumento **CadenaEntrada**.

Ejemplos

- **altova:last-chars**("2014-01-15", 5) devuelve 01-15-
- **altova:last-chars**("USA", 10) devuelve USA

pad-string-left [altova:]

altova:pad-string-left(CadenaParaRellenar *como xs:string*, **LongitudCadena** *como xs:integer*, **CarácterRelleno** *como xs:string***)** como **xs:string** XP3.1 XQ3.1
El argumento CarácterRelleno es un solo carácter. Se añade a la izquierda de la cadena para aumentar el número de caracteres de la CadenaParaRellenar, de modo que este número equivalga al valor entero del argumento LongitudCadena. El argumento LongitudCadena puede tener cualquier valor entero (positivo o negativo), pero el relleno solo se lleva a cabo si el valor de LongitudCadena es mayor que el número de caracteres de CadenaParaRellenar. Si CadenaParaRellenar tiene más caracteres que el valor de LongitudCadena, entonces CadenaParaRellenar se deja como está.

Ejemplos

- **altova:pad-string-left**('AP', 1, 'Z') devuelve 'AP'
- **altova:pad-string-left**('AP', 2, 'Z') devuelve 'AP'
- **altova:pad-string-left**('AP', 3, 'Z') devuelve 'ZAP'
- **altova:pad-string-left**('AP', 4, 'Z') devuelve 'ZZAP'
- **altova:pad-string-left**('AP', -3, 'Z') devuelve 'AP'
- **altova:pad-string-left**('AP', 3, 'YZ') devuelve un error indicando que el carácter de relleno es demasiado largo.

pad-string-right [altova:]

altova:pad-string-right(CadenaParaRellenar *como xs:string*, **LongitudCadena** *como xs:integer*, **CarácterRelleno** *como xs:string***)** como **xs:string** XP3.1 XQ3.1
El argumento CarácterRelleno es un solo carácter. Se añade a la derecha de la cadena para aumentar el número de caracteres de la CadenaParaRellenar, de modo que este número equivalga al valor entero del argumento LongitudCadena. El argumento LongitudCadena puede tener cualquier valor entero (positivo o negativo), pero el relleno solo se lleva a cabo si el valor de LongitudCadena es mayor que el número de caracteres de CadenaParaRellenar. Si CadenaParaRellenar tiene más caracteres que el valor de LongitudCadena, entonces CadenaParaRellenar se deja como está.

Ejemplos

- `altova:pad-string-right('AP', 1, 'Z')` devuelve `'AP'`
- `altova:pad-string-right('AP', 2, 'Z')` devuelve `'AP'`
- `altova:pad-string-right('AP', 3, 'Z')` devuelve `'APZ'`
- `altova:pad-string-right('AP', 4, 'Z')` devuelve `'APZZ'`
- `altova:pad-string-right('AP', -3, 'Z')` devuelve `'AP'`
- `altova:pad-string-right('AP', 3, 'YZ')` devuelve un error indicando que el carácter de relleno es demasiado largo.

repeat-string [altova:]

`altova:repeat-string(CadenaEntrada` *`as xs:string`*`, Repeticiones` *`as xs:integer`*`)` como **`xs:string`** **XP2** **XQ1** **XP3.1** **XQ3.1**

Genera una cadena que está compuesta por el primer argumento `CadenaEntrada` repetida tantas veces como indique el argumento `Repeticiones`.

Ejemplo

- **`altova:repeat-string`**`("Altova #", 3)`
  devuelve `Altova #Altova #Altova #`"

substring-after-last [altova:]

`altova:substring-after-last(CadenaPrincipal` *`as xs:string`*`, CadenaPrueba` *`as xs:string`*`)`**
como **`xs:string`** **XP3.1** **XQ3.1**

Si **`CadenaPrueba`** se encuentra en **`CadenaPrincipal`**, la función devuelve la subcadena que aparece después de **`CadenaPrueba`** en **`CadenaPrincipal`**. Si **`CadenaPrueba`** no está en **`CadenaPrincipal`**, entonces devuelve la cadena vacía. Si **`CadenaPrueba`** es una cadena vacía, entonces devuelve la **`CadenaPrincipal`** entera. Si **`CadenaPrueba`** aparece varias veces en **`CadenaPrincipal`**, la función devuelve la subcadena que aparece después de la última **`CadenaPrueba`**.

Ejemplos

- **`altova:substring-after-last`**`('ABCDEFGH', 'B')` devuelve `'CDEFGH'`
- **`altova:substring-after-last`**`('ABCDEFGH', 'BC')` devuelve `'DEFGH'`
- **`altova:substring-after-last`**`('ABCDEFGH', 'BD')` devuelve `''`
- **`altova:substring-after-last`**`('ABCDEFGH', 'Z')` devuelve `''`
- **`altova:substring-after-last`**`('ABCDEFGH', '')` devuelve `'ABCDEFGH'`
- **`altova:substring-after-last`**`('ABCD-ABCD', 'B')` devuelve `'CD'`
- **`altova:substring-after-last`**`('ABCD-ABCD-ABCD', 'BCD')` devuelve `''`

substring-before-last [altova:]

`altova:substring-before-last(CadenaPrincipal` *`as xs:string`*`, CadenaPrueba` *`as xs:string`*`)`**
como **`xs:string`** **XP3.1** **XQ3.1**

Si **`CadenaPrueba`** se encuentra en **`CadenaPrincipal`**, la función devuelve la subcadena que aparece después de **`CadenaPrueba`** en **`CadenaPrincipal`**. Si **`CadenaPrueba`** no está en **`CadenaPrincipal`**, entonces devuelve la cadena vacía. Si **`CadenaPrueba`** es una cadena vacía, entonces devuelve la **`CadenaPrincipal`** entera. Si **`CadenaPrueba`** aparece varias veces en **`CadenaPrincipal`**, la función

devuelve la subcadena que aparece antes de la última **CadenaPrueba**.

Ejemplos

- `altova:substring-before-last('ABCDEFGH', 'B')` devuelve `'A'`
- `altova:substring-before-last('ABCDEFGH', 'BC')` devuelve `'A'`
- `altova:substring-before-last('ABCDEFGH', 'BD')` devuelve `''`
- `altova:substring-before-last('ABCDEFGH', 'Z')` devuelve `''`
- `altova:substring-before-last('ABCDEFGH', '')` devuelve `''`
- `altova:substring-before-last('ABCD-ABCD', 'B')` devuelve `'ABCD-A'`
- `altova:substring-before-last('ABCD-ABCD-ABCD', 'ABCD')` devuelve `'ABCD-ABCD-'`

substring-pos [altova:]

altova:substring-pos(Cadena *as xs:string*, **CadenaBúsqueda** *as xs:string*) como **xs:integer** XP3.1 XQ3.1

Devuelve la posición de carácter de la primera instancia de **CadenaBúsqueda** en **Cadena**. La posición de carácter se devuelve como número entero. El primer carácter de **CadenaBúsqueda** tiene la posición 1. Si **CadenaBúsqueda** no aparece dentro de **Cadena**, la función devuelve el entero `0`. Para buscar la segunda instancia de **CadenaBúsqueda**, etc. use la otra firma de esta función.

Ejemplos

- `altova:substring-pos('Altova', 'to')` devuelve `3`
- `altova:substring-pos('Altova', 'tov')` devuelve `3`
- `altova:substring-pos('Altova', 'tv')` devuelve `0`
- `altova:substring-pos('AltovaAltova', 'to')` devuelve `3`

altova:substring-pos(Cadena *as xs:string*, **CadenaBúsqueda** *as xs:string*, **Entero** *as xs:integer*) como **xs:integer** XP3.1 XQ3.1

Devuelve la posición de carácter de **CadenaBúsqueda** en **Cadena**. La búsqueda de **CadenaBúsqueda** empieza en la posición de carácter dada por el argumento **Entero** (es decir, no se busca en la subcadena anterior a esta posición). El entero devuelto, sin embargo, es la posición que la cadena encontrada tiene en **Cadena**. Esta firma es muy práctica si quiere buscar la segunda posición, etc. de una cadena que aparece varias veces dentro de **Cadena**. Si **CadenaBúsqueda** no aparece en **Cadena**, la función devuelve el entero `0`.

Ejemplos

- `altova:substring-pos('Altova', 'to', 1)` devuelve `3`
- `altova:substring-pos('Altova', 'to', 3)` devuelve `3`
- `altova:substring-pos('Altova', 'to', 4)` devuelve `0`
- `altova:substring-pos('Altova-Altova', 'to', 0)` devuelve `3`
- `altova:substring-pos('Altova-Altova', 'to', 4)` devuelve `10`

substring-pos [altova:]

altova:substring-pos(Cadena *as xs:string*, **CadenaBúsqueda** *as xs:string*) como **xs:integer** XP3.1 XQ3.1

Devuelve la posición de carácter de la primera instancia de **CadenaBúsqueda** en **Cadena**. La posición de carácter se devuelve como número entero. El primer carácter de **CadenaBúsqueda** tiene la posición 1. Si **CadenaBúsqueda** no aparece dentro de **Cadena**, la función devuelve el entero `0`. Para buscar la segunda instancia de **CadenaBúsqueda**, etc. use la otra firma de esta función.

Ejemplos

- `altova:substring-pos('Altova', 'to')` devuelve `3`
- `altova:substring-pos('Altova', 'tov')` devuelve `3`
- `altova:substring-pos('Altova', 'tv')` devuelve `0`
- `altova:substring-pos('AltovaAltova', 'to')` devuelve `3`

altova:substring-pos(Cadena *as xs:string*, **CadenaBúsqueda** *as xs:string*, **Entero** *as xs:integer***)** como **xs:integer** XP3.1 XQ3.1
Devuelve la posición de carácter de **CadenaBúsqueda** en **Cadena**. La búsqueda de **CadenaBúsqueda** empieza en la posición de carácter dada por el argumento **Entero** (es decir, no se busca en la subcadena anterior a esta posición). El entero devuelto, sin embargo, es la posición que la cadena encontrada tiene en **Cadena**. Esta firma es muy práctica si quiere buscar la segunda posición, etc. de una cadena que aparece varias veces dentro de **Cadena**. Si **CadenaBúsqueda** no aparece en **Cadena**, la función devuelve el entero `0`.

Ejemplos

- `altova:substring-pos('Altova', 'to', 1)` devuelve `3`
- `altova:substring-pos('Altova', 'to', 3)` devuelve `3`
- `altova:substring-pos('Altova', 'to', 4)` devuelve `0`
- `altova:substring-pos('Altova-Altova', 'to', 0)` devuelve `3`
- `altova:substring-pos('Altova-Altova', 'to', 4)` devuelve `10`

trim-string [altova:]

altova:trim-string(CadenaEntrada *as xs:string***)** como **xs:string** XP3.1 XQ3.1
Esta función toma un argumento `xs:string`, quita los espacios en blanco iniciales y finales y devuelve un `xs:string` "recortado".

Ejemplos

- `altova:trim-string("  Hello World  ")` devuelve `"Hello World"`
- `altova:trim-string("Hello World  ")` devuelve `"Hello World"`
- `altova:trim-string("  Hello World")` devuelve `"Hello World"`
- `altova:trim-string("Hello World")` devuelve `"Hello World"`
- `altova:trim-string("Hello  World")` devuelve `"Hello  World"`

trim-string-left [altova:]

altova:trim-string-left(CadenaEntrada *as xs:string***)** como **xs:string** XP3.1 XQ3.1
Esta función toma un argumento `xs:string`, quita los espacios en blanco iniciales y devuelve un `xs:string` recortado por la izquierda.

Ejemplos

- `altova:trim-string-left("  Hello World  ")` devuelve `"Hello World  "`
- `altova:trim-string-left("Hello World  ")` devuelve `"Hello World  "`
- `altova:trim-string-left("  Hello World")` devuelve `"Hello World"`
- `altova:trim-string-left("Hello World")` devuelve `"Hello World"`
- `altova:trim-string-left("Hello  World")` devuelve `"Hello  World"`

▼ trim-string-right [altova:]

altova:trim-string-right(CadenaEntrada *as xs:string***)** como **xs:string** XP3.1 XQ3.1
Esta función toma un argumento `xs:string`, quita los espacios en blanco finales y devuelve una cadena `xs:string` recortada por la derecha.

⊟ *Ejemplos*

- **altova:trim-string-right**(" Hello World ")) devuelve " Hello World"
- **altova:trim-string-right**("Hello World ")) devuelve "Hello World"
- **altova:trim-string-right**(" Hello World")) devuelve " Hello World"
- **altova:trim-string-right**("Hello World")) devuelve "Hello World"
- **altova:trim-string-right**("Hello World")) devuelve "Hello World"

12.2.2.1.9 Funciones XPath/XQuery: Varias

Estas funciones de extensión XPath/XQuery generales son compatibles con la versión actual de MapForce y se pueden usar en (i) expresiones XPath en contextos XSLT o (ii) en expresiones XQuery en documentos XQuery.

Nota sobre el nombre de las funciones y lenguajes

Puede utilizar todas las funciones de extensión de Altova en sus expresiones XPath/XQuery. Con ellas conseguirá funciones adicionales no disponibles en la biblioteca de funciones estándar de XPath, XQuery y XSLT. Las funciones de extensión de Altova están en el **espacio de nombres** **http://www.altova.com/xslt-extensions** y en esta sección se presentan con el prefijo, que se supone estará enlazado al espacio de nombres señalado. Tenga en cuenta que en futuras versiones del producto algunas funciones pueden dejar de ser compatibles o su comportamiento puede cambiar. Por tanto, consulte siempre la documentación del producto para conocer el funcionamiento de estas funciones en cada versión del producto.

Funciones XPath (en expresiones XPath en XSLT):	XP1 XP2 XP3.1.1
Funciones XSLT (en expresiones XPath en XSLT):	XSLT1 XSLT2 XSLT3
Funciones XQuery (en expresiones XQuery en XQuery):	XQ1 XQ3.1

▼ decode-string [altova:]

altova:decode-string(Input *as xs:base64Binary***) como xs:string** XP3.1 XQ3.1
altova:decode-string(Input *as xs:base64Binary***, Encoding como** *xs:string***) como xs:string** XP3.1 XQ3.1
Descifra la entrada en base64Binary en una cadena con el cifrado que se indique. Si no se indica ninguno se usa UTF-8. Estos son los cifrados compatibles: `US-ASCII, ISO-8859-1, UTF-16, UTF-16LE, UTF-16BE, ISO-10646-UCS2, UTF-32, UTF-32LE, UTF-32BE, ISO-10646-UCS4`

⊟ *Ejemplos*

- **altova:decode-string**(`$XML1/MailData/Meta/b64B`) devuelve la entrada en base64Binary como cadena de texto cifrada en UTF-8
- **altova:decode-string**(`$XML1/MailData/Meta/b64B, "UTF-8"`) devuelve la entrada en base64Binary como cadena de texto cifrada en UTF-8
- **altova:decode-string**(`$XML1/MailData/Meta/b64B, "ISO-8859-1"`) devuelve la entrada en base64Binary como una cadena de texto cifrada en ISO-8859-1

▼ encode-string [altova:]

altova:encode-string(InputString *como xs:string***) como xs:base64Binaryinteger** XP3.1 XQ3.1
altova:encode-string(InputString *como xs:string***, Encoding** *como xs:string***) como xs:base64Binaryinteger** XP3.1 XQ3.1
Cifra una cadena de texto usando el cifrado que se indique. Si no se indica ninguno, entonces se usa UTF-8. La cadena cifrada se convierte en caracteres base64Binary y se devuelve el valor base64Binary convertido. De momento se admite UTF-8, pero ampliaremos la compatibilidad a: `US-ASCII, ISO-8859-1, UTF-16, UTF-16LE, UTF-16BE, ISO-10646-UCS2, UTF-32, UTF-32LE, UTF-32BE, ISO-10646-UCS4`

▣ *Ejemplos*

- **altova:encode-string**(`"Altova"`) devuelve el equivalente en base64Binary de la cadena de texto cifrada en UTF-8 `"Altova"`
- **altova:encode-string**(`"Altova", "UTF-8"`) devuelve el equivalente en base64Binary de la cadena de texto cifrada en UTF-8 `"Altova"`

▼ get-temp-folder [altova:]

altova:get-temp-folder() como `xs:string` XP2 XQ1 XP3.1 XQ3.1
Esta función no toma ningún argumento. Devuelve la ruta de acceso de la carpeta temporal del usuario actual.

▣ *Ejemplo*

- **altova:get-temp-folder**()
en un equipo Windows devuelve (más o menos) `C:\Usuarios\<Usuario>\AppData\Local\Temp\` como valor de tipo `xs:string`.

▼ generate-guid [altova:]

altova:generate-guid() asxs:string XP2 XQ1 XP3.1 XQ3.1
Genera una cadena única de la interfaz gráfica del usuario.
⊞ *Ejemplo*

- **altova:generate-guid**() devuelve (por ejemplo) `85F971DA-17F3-4E4E-994E-99137873ACCD`

▼ high-res-timer [altova:]

altova:high-res-timer() como **xs:double** XP3.1 XQ3.1
Devuelve un valor de temporizador de alta resolución en segundos. La presencia de un temporizador de

alta resolución en un sistema permite hacer mediciones de alta precisión si es necesario (por ejemplo, en animaciones y para precisar de forma exacta horas de ejecución de código). Esta función ofrece la resolución del temporizador de alta resolución del sistema.

Ejemplos

- `altova:high-res-timer()` devuelve algo como `'1.16766146154566E6'`

parse-html [altova:]

`altova:parse-html(HTMLText` *`as xs:string`***`)`** como **`node()`** XP3.1 XQ3.1
El argumento `HTMLText` es una cadena que contiene el texto de un documento HTML. La función crea una estructura HTML a partir de la cadena. La cadena enviada puede contener o no el elemento HTML. En ambos casos el elemento raíz de la estructura es un elemento llamado **`HTML`**. Asegúrese de que el código HTML de la cadena enviada es válido.

Ejemplos

- `altova:parse-html("<html><head/><body><h1>Header</h1></body></html>")` crea una estructura HTML a partir de la cadena enviada

sleep [altova:]

`altova:sleep(Millisecs` *`como xs:integer`***`)`** como **`empty-sequence()`** XP2 XQ1 XP3.1 XQ3.1
Suspende la ejecución de la operación actual durante el número de milisegundos dado por el argumento `Millisecs`.

Ejemplos

- `altova:sleep(1000)` suspende la ejecución de la operación actual durante 1000 millisegundos.

[Subir 574]

12.2.2.2 Funciones de extensión varias

Los lenguajes de programación como Java y C# ofrecen varias funciones predefinidas que no están disponibles como funciones XQuery/XPath ni XSLT. Un ejemplo son las funciones matemáticas de Java `sin()` y `cos()`. Si los diseñadores de hojas de estilos XSLT y consultas XQuery tuvieran acceso a estas funciones, el área de aplicación de sus hojas de estilos y consultas aumentaría y su trabajo sería un poco más sencillo.

Los motores XSLT y XQuery de los productos de Altova admiten el uso de funciones de extensión en Java 577 y .NET 586, así como scripts MSXSL para XSLT 592.

Esta sección describe cómo usar funciones de extensión y scripts MSXSL en hojas de estilos XSLT y documentos XQuery. Las funciones de extensión pueden organizarse en varios grupos:

- Funciones de extensión Java 577
- Funciones de extensión .NET 586

- Scripts MSXSL para XSLT 592

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.

12.2.2.2.1 Funciones de extensión Java

Puede usar una función de extensión Java dentro de una expresión XPath o XQuery para invocar un constructor Java o llamar a un método Java (estático o de instancia).

Un campo de una clase Java se trata como un método sin argumentos. Un campo puede ser estático o de instancia. Más adelante describimos cómo se accede a los campos estáticos y de instancia.

Este apartado tiene varias partes:

- Archivos de clases definidos por el usuario 579
- Archivos JAR definidos por el usuario 582
- Java: Constructores 583
- Java: Métodos estáticos y campos estáticos 583
- Java: Métodos de instancia y campos de instancia 584
- Tipos de datos: Conversión de XPath/XQuery en Java 585
- Tipos de datos: Conversión de Java en XPath/XQuery 586

Tenga en cuenta que:

- Si está usando un producto de escritorio de Altova, la aplicación intentará detectar automáticamente la ruta de acceso al equipo virtual Java; para ello leerá (en este orden): (i) el registro de Windows y (ii) la variable de entorno **JAVA_HOME**. También puede añadir una ruta personal en el cuadro de diálogo "Opciones" de la aplicación; esta ruta tendrá prioridad frente a cualquier otra ruta de acceso a un equipo virtual Java que se detecte automáticamente.
- Si está usando un producto servidor de Altova en un equipo Windows, la ruta de acceso al equipo virtual Java se leerá primero desde el registro de Windows; si esto no ocurre se usa la variable de entorno **JAVA_HOME**.
- Si está usando un producto servidor de Altova en un equipo Linux o macOS, entonces asegúrese de que la variable de entorno **JAVA_HOME** está definida correctamente y la biblioteca Java de equipos virtuales (en Windows, el archivo **jvm.dll**) se encuentra en uno de estos directorios: **\bin\server** o **\bin\client**.

Formato de la función de extensión

La función de extensión de la expresión XPath/XQuery debe tener este formato `prefijo:nombreFunción()`.

- La parte `prefijo:` identifica la función de extensión como función Java. Lo hace asociando la función de extensión con una declaración de espacio de nombres del ámbito, cuyo URI debe empezar por `java:` (*ver ejemplos más abajo*). La declaración de espacio de nombres debe identificar una clase Java, por ejemplo: `xmlns:myns="java:java.lang.Math"`. Sin embargo, también puede ser simplemente: `xmlns:myns="java"` (sin los dos puntos), dejando la identificación de la clase Java a la parte `nombreFunción()` de la función de extensión.
- La parte `nombreFunción()`identifica el método Java al que se llama y presenta los argumentos para el método (*ver ejemplos más abajo*). Sin embargo, si el URI de espacio de nombres identificado por la parte `prefijo:` no identifica una clase Java (*ver punto anterior*), entonces la clase Java debe identificarse en la parte `nombreFunción()`, antes de la clase y separada de la clase por un punto (*ver el segundo ejemplo XSLT que aparece más abajo*).

Nota: La clase a la que se llama debe estar en la ruta de acceso de clase del equipo.

Ejemplo de código XSLT

Aquí ofrecemos dos ejemplos de cómo se puede llamar a un método estático. En el primer ejemplo, el nombre de la clase (`java.lang.Math`) se incluye en el URI de espacio de nombres y, por tanto, no puede estar en la parte `nombreFunción()`. En el segundo ejemplo, la parte `prefijo:` presenta el prefijo `java:` mientras que la parte `nombreFunción()` identifica la clase y el método.

```
<xsl:value-of xmlns:jMath="java:java.lang.Math"
              select="jMath:cos(3.14)" />

<xsl:value-of xmlns:jmath="java"
              select="jmath:java.lang.Math.cos(3.14)" />
```

El método nombrado en la función de extensión (`cos()`) debe coincidir con el nombre de un método estático público de la clase Java nombrada (`java.lang.Math`).

Ejemplo de código XQuery

Aquí puede ver un ejemplo de código XQuery similar al código XSLT anterior:

```
<cosine xmlns:jMath="java:java.lang.Math">
   {jMath:cos(3.14)}
</cosine>
```

Clases Java definidas por el usuario

Si creó sus propias clases Java, a los métodos de estas clases se les llama de otra manera, dependiendo de: (i) si a las clases se accede por medio de un archivo JAR o de un archivo de clases y (ii) si estos archivos están en el directorio actual (el directorio del documento XSLT o XQuery). Para más información consulte los apartados Archivos de clases definidos por el usuario 579 y Archivos Jar definidos por el usuario 582. 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.

12.2.2.2.1.1 Archivos de clases definidos por el usuario

Si se accede a las clases por medio de un archivo de clases, entonces hay cuatro posibilidades:

- El archivo de clases está en un paquete. El archivo XSLT/XQuery está en la misma carpeta que el paquete Java. (*ver ejemplo* 579)
- 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* 580)
- El archivo de clases está en un paquete. El archivo XSLT/XQuery está en una carpeta cualquiera. (*ver ejemplo* 580)
- El archivo de clases no está en un paquete. El archivo XSLT/XQuery está una carpeta cualquiera. (*ver ejemplo* 581)

Imaginemos que tenemos un archivo de clases que no está en un paquete y que está en la misma carpeta que el documento XSLT/XQuery. En este caso, puesto que en la carpeta se encuentran todas las clases, no es necesario especificar la ubicación del archivo. La sintaxis que se utiliza para identificar una clase es esta:

```
java:nombreClase
```

donde

`java:` indica que se está llamando a una función definida por el usuario (por defecto se cargan las clases Java del directorio actual)
`nombreClase` es el nombre de la clase del método elegido

La clase se identifica en un URI de espacio de nombres y el espacio de nombres se usa como prefijo para la llamada al método.

El archivo de clases está en un paquete. El archivo XSLT/XQuery está en la misma carpeta que el paquete Java

El código que aparece a continuación llama al método `getVehicleType()` de la clase `Car` del paquete `com.altova.extfunc`. El paquete `com.altova.extfunc` está en la carpeta `JavaProject`. El archivo XSLT también está en la carpeta `JavaProject`.

```
<xsl:stylesheet version="2.0"
       xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
       xmlns:xs="http://www.w3.org/2001/XMLSchema"
       xmlns:fn="http://www.w3.org/2005/xpath-functions"
       xmlns:car="java:com.altova.extfunc.Car" >
<xsl:output exclude-result-prefixes="fn car xsl fo xs"/>

<xsl:template match="/">
   <a>
   <xsl:value-of select="car:getVehicleType()"/>
   </a>
</xsl:template>

</xsl:stylesheet>
```

El archivo de clases está referenciado. El archivo XSLT/XQuery está en la misma carpeta que el archivo de clases

El código que aparece a continuación llama al método `getVehicleType()` de la clase `Car`. Digamos que: (i) el archivo de clases `Car` está en esta carpeta: `JavaProject/com/altova/extfunc` y que (ii) esa carpeta es la del ejemplo siguiente. El archivo XSLT también está en la carpeta `JavaProject/com/altova/extfunc`.

```
<xsl:stylesheet version="2.0"
       xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
       xmlns:xs="http://www.w3.org/2001/XMLSchema"
       xmlns:fn="http://www.w3.org/2005/xpath-functions"
       xmlns:car="java:Car" >
<xsl:output exclude-result-prefixes="fn car xsl fo xs"/>

<xsl:template match="/">
   <a>
   <xsl:value-of select="car:getVehicleType()"/>
   </a>
</xsl:template>

</xsl:stylesheet>
```

El archivo de clases está en un paquete. El archivo XSLT/XQuery está en una carpeta cualquiera

El código que aparece a continuación llama al método `getCarColor()` de la clase `Car` del paquete `com.altova.extfunc`. El paquete `com.altova.extfunc` está en la carpeta `JavaProject`. El archivo XSLT está en otra carpeta cualquiera. En este caso debe especificarse la ubicación del paquete dentro del URI como una cadena de consulta. La sintaxis es esta:

```
java:nombreClase[?ruta=uri-del-paquete]
```

donde

`java`: indica que se está llamando a una función Java definida por el usuario
`uri-del-paquete` es el URI del paquete Java
`nombreClase` es el nombre de la clase del método elegido

La clase se identifica en un URI de espacio de nombres y el espacio de nombres se usa como prefijo para la llamada al método. El ejemplo de código que aparece a continuación explica cómo se accede a un archivo de clases que está ubicado en un directorio que no es el directorio actual.

```
<xsl:stylesheet version="2.0"
       xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
       xmlns:xs="http://www.w3.org/2001/XMLSchema"
       xmlns:fn="http://www.w3.org/2005/xpath-functions"
       xmlns:car="java:com.altova.extfunc.Car?path=file:///C:/JavaProject/" >

<xsl:output exclude-result-prefixes="fn car xsl xs"/>
```

```
<xsl:template match="/">
   <xsl:variable name="myCar" select="car:new('red')" />
   <a><xsl:value-of select="car:getCarColor($myCar)"/></a>
</xsl:template>

</xsl:stylesheet>
```

El archivo de clases no está en un paquete. El archivo XSLT/XQuery está una carpeta cualquiera

El código que aparece a continuación llama al método `getCarColor()`de la clase `Car`. Digamos que el archivo de clases `Car` está en la carpeta `C:/JavaProject/com/altova/extfunc` y que el archivo XSLT está en otra carpeta cualquiera. En este caso debe especificarse la ubicación del paquete dentro del URI como una cadena de consulta. La sintaxis es esta:

```
java:nombreClase[?ruta=<uri-del-archivoClases>]
```

donde

`java`: indica que se está llamando a una función Java definida por el usuario
`uri-del-archivoClases` es el URI de la carpeta donde se ubica el archivo de clases
`nombreClase` es el nombre de la clase del método elegido

La clase se identifica en un URI de espacio de nombres y el espacio de nombres se usa como prefijo para la llamada al método. El ejemplo de código que aparece a continuación explica cómo se accede a un archivo de clases que está ubicado en un directorio que no es el directorio actual.

```
<xsl:stylesheet version="2.0"
       xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
       xmlns:xs="http://www.w3.org/2001/XMLSchema"
       xmlns:fn="http://www.w3.org/2005/xpath-functions"
       xmlns:car="java:Car?path=file:///C:/JavaProject/com/altova/extfunc/" >

<xsl:output exclude-result-prefixes="fn car xsl xs"/>

<xsl:template match="/">
   <xsl:variable name="myCar" select="car:new('red')" />
   <a><xsl:value-of select="car:getCarColor($myCar)"/></a>
</xsl:template>

</xsl:stylesheet>
```

Nota: Cuando se presenta una ruta de acceso por medio de una función de extensión, la ruta de acceso se añade al ClassLoader.

12.2.2.2.1.2 Archivos JAR definidos por el usuario

Si se accede a las clases por medio de un archivo JAR, entonces se debe especificar el URI del archivo JAR usando esta sintaxis:

```
xmlns:claseEspacioNombres="java:nombreClase?ruta=jar:uri-del-archivoJar!/"
```

Para la llamada al método se usa el prefijo del URI de espacio de nombres que identifica la clase: `claseEspacioNombres:método()`

En la sintaxis anterior:

`java:` indica que se está llamando a una función de Java
`nombreClase` es el nombre de la clase definida por el usuario
`?` es el separador entre el nombre de la clase y la ruta de acceso
`ruta=jar:` indica que se ofrece una ruta de acceso a un archivo JAR
`uri-del-archivoJar` es el URI del archivo JAR
`!/` 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.

12.2.2.2.1.3 Java: Constructores

Una función de extensión se puede usar para llamar a un constructor Java. A todos los constructores se les llama con la pseudofunción `new()`.

Si el resultado de una llamada a un constructor Java se puede convertir de manera implícita a tipos de datos XPath/XQuery [586], 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* [584]):
  ```
  <xsl:value-of select="date:toString(date:new())" xmlns:date="java:java.util.Date" />
  ```

12.2.2.2.1.4 Java: Métodos estáticos y campos estáticos

La llamada a un método estático la hace directamente su nombre Java y se hace presentando los argumentos para el método. A los campos estáticos (es decir, los métodos que no toman argumentos), como los campos de valor constante `E` y `PI`, se accede sin especificar ningún argumento.

Ejemplos de código XSLT

Aquí puede ver varios ejemplos de cómo se llama a métodos y campos estáticos:

```
<xsl:value-of xmlns:jMath="java:java.lang.Math"
              select="jMath:cos(3.14)" />

<xsl:value-of xmlns:jMath="java:java.lang.Math"
              select="jMath:cos( jMath:PI() )" />

<xsl:value-of xmlns:jMath="java:java.lang.Math"
              select="jMath:E() * jMath:cos(3.14)" />
```

Observe que las funciones de extensión anteriores tienen el formato `prefijo:nombreFunción()`. En los tres ejemplos anteriores, el prefijo es `jMath:`, que está asociado al URI de espacio de nombres `java:java.lang.Math`. (El URI de espacio de nombres debe empezar por `java:`. En los ejemplos anteriores se extiende para contener el nombre de la clase (`java.lang.Math`).) La parte `nombreFunción()` de las funciones de extensión debe coincidir con el nombre de una clase pública (p. ej. `java.lang.Math`) seguido del

nombre de un método estático público con sus argumentos (como `cos(3.14)`) o de un campo estático público (como `PI()`).

En los tres ejemplos anteriores, el nombre de la clase se incluyó en el URI de espacio de nombres. Si no estuviera en el URI de espacio de nombres, se incluiría en la parte `nombreFunción()` de la función de extensión. Por ejemplo:

```
<xsl:value-of xmlns:java="java:"
              select="java:java.lang.Math.cos(3.14)" />
```

Ejemplo de XQuery

Un ejemplo de XQuery similar sería:

```
<cosine xmlns:jMath="java:java.lang.Math">
   {jMath:cos(3.14)}
</cosine>
```

12.2.2.2.1.5 Java: Métodos de instancia y campos de instancia

A un método de instancia se le pasa un objeto Java como primer argumento de la llamada a método. Dicho objeto Java suele crearse usando una función de extensión (por ejemplo, una llamada a un constructor) o un parámetro o una variable de hoja de estilos. Un ejemplo de código XSLT de este tipo sería:

```
<xsl:stylesheet version="1.0" exclude-result-prefixes="date"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:date="java:java.util.Date"
  xmlns:jlang="java:java.lang">
  <xsl:param name="CurrentDate" select="date:new()"/>
  <xsl:template match="/">
      <enrollment institution-id="Altova School"
                date="{date:toString($CurrentDate)}"
                type="{jlang:Object.toString(jlang:Object.getClass( date:new() ))}">
      </enrollment>
  </xsl:template>
</xsl:stylesheet>
```

En el ejemplo anterior el valor del nodo `enrollment/@type` se crea de la siguiente manera:

1. Se crea un objeto con un constructor para la clase `java.util.Date` (con el constructor `date:new()`).
2. Este objeto Java se pasa como argumento del método `jlang.Object.getClass`.
3. El objeto que obtiene el método `getClass` se pasa como argumento al método `jlang.Object.toString`.

El resultado (el valor de `@type`) será una cadena con este valor: `java.util.Date`.

En teoría, un campo de instancia es diferente de un método de instancia porque al campo de instancia no se pasa como argumento un objeto Java propiamente dicho. En su lugar se pasa como argumento un parámetro o variable. Sin embargo, el parámetro o la variable puede contener el valor devuelto por un objeto Java. Por ejemplo, el parámetro `CurrentDate` toma el valor que devolvió un constructor para la clase `java.util.Date`.

Este valor se pasa después como argumento al método de instancia `date:toString` a fin de suministrar el valor de `/enrollment/@date`.

12.2.2.2.1.6 Tipos de datos: Conversión de XPath/XQuery en Java

Cuando se llama a una función Java desde dentro de una expresión XPath/XQuery, el tipo de datos de los argumentos de la función es importante a la hora de determinar a cuál de las clases Java que tienen el mismo nombre se llama.

En Java se siguen estas reglas:

- Si hay más de un método Java con el mismo nombre, pero cada método tiene un número diferente de argumentos, entonces se selecciona el método Java que mejor se ajusta al número de argumentos de la llamada a función.
- Los tipos de datos de cadena, numéricos y booleanos de XPath/XQuery (*ver lista más abajo*) se convierten de forma implícita en el tipo de datos Java correspondiente. Si el tipo XPath/XQuery suministrado se puede convertir a más de un tipo Java (p. ej. `xs:integer`), entonces se selecciona el tipo Java que se declaró para el método seleccionado. Por ejemplo, si el método Java al que se llama es `fx(decimal)` y el tipo de datos XPath/XQuery suministrado es `xs:integer`, entonces `xs:integer` se convierte en el tipo de datos Java `decimal`.

La tabla que aparece a continuación enumera las conversiones implícitas de los tipos de cadena, numéricos y booleanos XPath/XQuery en tipos de datos Java.

`xs:string`	`java.lang.String`
`xs:boolean`	`boolean` (primitivo), `java.lang.Boolean`
`xs:integer`	int, long, short, byte, float, double y sus clases contenedoras, como `java.lang.Integer`
`xs:float`	`float` (primitivo), `java.lang.Float`, `double` (primitivo)
`xs:double`	`double` (primitivo), `java.lang.Double`
`xs:decimal`	`float` (primitivo), `java.lang.Float`, `double`(primitivo), `java.lang.Double`

Los subtipos de los tipos de datos XML Schema de la tabla anterior (que se usan en XPath y XQuery) también se convierten en los tipos Java correspondientes al tipo antecesor del subtipo.

En algunos casos quizás no sea posible seleccionar el método Java correcto usando la información dada. Por ejemplo, imagine que:

- El argumento presentado es un valor `xs:untypedAtomic` de 10 y está destinado al método `mimétodo(float)`.
- Sin embargo, hay otro método en la clase que toma un argumento de otro tipo de datos: `mimétodo(double)`.

- Puesto que los métodos tienen el mismo nombre y el tipo suministrado (`xs:untypedAtomic`) se puede convertir correctamente tanto en `float` como en `double`, es posible que `xs:untypedAtomic` se convierta en `double` en lugar de en `float`.
- Por consiguiente, el método seleccionado no será el método necesario y quizás no produzca el resultado esperado. Una solución es crear un método definido por el usuario con un nombre diferente y usar ese método.

Los tipos que no aparecen en la lista anterior (p. ej. `xs:date`) no se convertirán y generarán un error. No obstante, tenga en cuenta que en algunos casos, es posible crear el tipo Java necesario usando un constructor Java.

12.2.2.2.1.7 Tipos de datos: Conversión de Java en XPath/XQuery

Cuando un método Java devuelve un valor y el tipo de datos del valor es un tipo de cadena, numérico o booleano, entonces se convierte en el tipo de datos XPath/XQuery correspondiente. Por ejemplo, los tipos de datos Java `java.lang.Boolean` y `boolean` se convierten en `xsd:boolean`.

Las matrices unidimensionales devueltas por las funciones se extienden en una secuencia. Las matrices multidimensionales no se convierten y, por tanto, deberían ser contenidas.

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).

12.2.2.2.2 Funciones de extensión .NET

Si trabaja en la plataforma .NET desde un equipo Windows, puede usar funciones de extensión escritas en cualquier lenguaje .NET (p. ej. C#). Una función de extensión .NET se puede usar dentro de una expresión XPath/XQuery para invocar un constructor, una propiedad o un método (estático o de instancia) de una clase .NET.

A una propiedad de una clase .NET se le llama usando la sintaxis `get_NombrePropiedad()`.

Este apartado tiene varias partes:

Formato de la función de extensión

La función de extensión de la expresión XPath/XQuery debe tener este formato `prefijo:nombreFunción()`.

- La parte `prefijo:` está asociada a un URI que identifica la clase .NET.
- La parte `nombreFunción()` identifica el constructor, la propiedad o el método (estático o de instancia) dentro de la clase .NET y, si es necesario, suministra los argumentos.
- El URI debe empezar por `clitype:` (que identifica la función como función de extensión .NET).
- El formato `prefijo:nombreFunción()` de la función de extensión se puede usar con clases del sistema y con clases de un ensamblado cargado. No obstante, si se tiene que cargar una clase, será necesario suministrar parámetros que contengan la información necesaria.

Parámetros

Para cargar un ensamblado se usan estos parámetros:

`asm`	El nombre del ensamblado que se debe cargar.
`ver`	El número de versión (máximo cuatro enteros separados por puntos).
`sn`	El símbolo de clave del nombre seguro del ensamblado (16 dígitos hexadecimales).
`from`	Un URI que da la ubicación del ensamblado (DLL) que se debe cargar. Si el URI es relativo, es relativo al archivo XSLT o XQuery. Si está presente este parámetro, se ignoran los demás parámetros.
`partialname`	El nombre parcial del ensamblado. Se suministra a `Assembly.LoadWith.PartialName()`, que intentará cargar el ensamblado. Si está presente el parámetro `partialname`, se ignoran los demás parámetros.
`loc`	La configuración regional, por ejemplo, `en-US`. La configuración predeterminada es `neutral`.

Si el ensamblado se debe cargar desde un archivo DLL, use el parámetro `from` y omita el parámetro `sn`. Si el ensamblado se debe cargar desde el caché general de ensamblados (GAC), use el parámetro `sn` y omita el parámetro `from`.

Debe insertar un signo de interrogación final antes del primer parámetro y los parámetros deben separarse con un punto y coma (;). El nombre de parámetro da su valor con un signo igual (=), como en el ejemplo que aparece más abajo.

Ejemplos de declaraciones de espacios de nombres

Esto es un ejemplo de una declaración de espacio de nombres en XSLT que identifica la clase del sistema `System.Environment`:

```
xmlns:myns="clitype:System.Environment"
```

Esto es un ejemplo de una declaración de espacio de nombres en XSLT que identifica la clase que se debe cargar como `Trade.Forward.Scrip`:

```
xmlns:myns="clitype:Trade.Forward.Scrip?asm=forward;version=10.6.2.1"
```

Esto es un ejemplo de una declaración de espacio de nombres en XQuery que identifica la clase del sistema `MyManagedDLL.testClass:`. Existen dos tipos de clases:

1. Cuando el ensamblado se carga desde el GAC:
   ```
   declare namespace cs="clitype:MyManagedDLL.testClass?asm=MyManagedDLL;
   ver=1.2.3.4;loc=neutral;sn=b9f091b72dccfba8";
   ```

2. Cuando el ensamblado se carga desde el archivo DLL (ver las referencias parciales y completas):
   ```
   declare namespace cs="clitype:MyManagedDLL.testClass?from=file:///C:/Altova
   Projects/extFunctions/MyManagedDLL.dll;

   declare namespace cs="clitype:MyManagedDLL.testClass?from=MyManagedDLL.dll;
   ```

Ejemplo de código XSLT

Aquí puede ver un ejemplo de código XSLT que llama a funciones de la clase del sistema `System.Math`:

```
<xsl:stylesheet version="2.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:fn="http://www.w3.org/2005/xpath-functions">
   <xsl:output method="xml" omit-xml-declaration="yes" />
   <xsl:template match="/">
      <math xmlns:math="clitype:System.Math">
         <sqrt><xsl:value-of select="math:Sqrt(9)"/></sqrt>
         <pi><xsl:value-of select="math:PI()"/></pi>
         <e><xsl:value-of select="math:E()"/></e>
         <pow><xsl:value-of select="math:Pow(math:PI(), math:E())"/></pow>
      </math>
   </xsl:template>
</xsl:stylesheet>
```

La declaración de espacio de nombres del elemento `math` asocia el prefijo `math:` al URI `clitype:System.Math`. La parte inicial `clitype:` del URI indica que lo que sigue identifica una clase del sistema o una clase cargada. El prefijo `math:` de las expresiones XPath asocia las funciones de extensión al URI (y, por extensión, a la clase) `System.Math`. Las funciones de extensión identifican métodos en la clase `System.Math` y presenta argumentos cuando es necesario.

Ejemplo de código XQuery

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.

12.2.2.2.2.1 .NET: Constructores

Una función de extensión se puede usar para llamar a un constructor .NET. A todos los constructores se les llama con la pseudofunción `new()`. Si hay más de un constructor para una clase, entonces se selecciona el constructor que más se ajusta al número de argumentos suministrados. Si no se encuentra ningún constructor que coincida con los argumentos suministrados, entonces se genera el error "`No constructor found`".

Constructores que devuelven tipos de datos XPath/XQuery

Si el resultado de una llamada a un constructor .NET se puede convertir de forma implícita en tipos de datos XPath/XQuery [586], 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* [584]):
```
<xsl:value-of select="date:ToString(date:new(2008, 4, 29))"
xmlns:date="clitype:System.DateTime" />
```
- Se puede convertir en un tipo de cadena, numérico o booleano:
```
<xsl:value-of select="xs:integer(date:get_Month(date:new(2008, 4, 29)))"
xmlns:date="clitype:System.DateTime" />
```

12.2.2.2.2.2 .NET: Metodos estáticos y campos estáticos

La llamada a un método estático la hace directamente su nombre y se hace presentando los argumentos para el método. El nombre usado en la llamada debe ser el mismo que un método estático público de la clase especificada. Si el nombre del método y el número de argumentos que se dio en la llamada a función coincide con algún método de la clase, entonces los tipos de los argumentos presentados se evalúan para encontrar el resultado ideal. Si no se encuentra ninguna coincidencia, se emite un error.

Nota: Un campo de una clase .NET se trata como si fuera un método sin argumentos. Para llamar a una propiedad se usa la sintaxis `get_nombrePropiedad()`.

Ejemplos

Este ejemplo de código XSLT muestra una llamada a un método con un argumento (`System.Math.Sin(arg)`):

```
<xsl:value-of select="math:Sin(30)" xmlns:math="clitype:System.Math"/>
```

Este ejemplo de código XSLT muestra una llamada a un campo (que se trata como si fuera un método sin argumentos) (`System.Double.MaxValue()`):

```
<xsl:value-of select="double:MaxValue()" xmlns:double="clitype:System.Double"/>
```

Este ejemplo de código XSLT muestra una llamada a una propiedad (la sintaxis es `get_nombrePropiedad()`) (`System.String()`):

```
<xsl:value-of select="string:get_Length('my string')"
xmlns:string="clitype:System.String"/>
```

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>
```

12.2.2.2.2.3 .NET: Métodos de instancia y campos de instancia

Un método de instancia es un método al que se le pasa un objeto .NET como primer argumento de la llamada al método. Este objeto .NET se suele crear usando una función de extensión (por ejemplo, una llamada a un constructor) o un parámetro o una variable de una hoja de estilos. Un ejemplo de código XSLT para este tipo de método sería:

```
<xsl:stylesheet version="2.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:fn="http://www.w3.org/2005/xpath-functions">
   <xsl:output method="xml" omit-xml-declaration="yes"/>
   <xsl:template match="/">
      <xsl:variable name="releasedate"
         select="date:new(2008, 4, 29)"
         xmlns:date="clitype:System.DateTime"/>
      <doc>
         <date>
            <xsl:value-of select="date:ToString(date:new(2008, 4, 29))"
               xmlns:date="clitype:System.DateTime"/>
         </date>
         <date>
            <xsl:value-of select="date:ToString($releasedate)"
```

```
                xmlns:date="clitype:System.DateTime"/>
            </date>
        </doc>
    </xsl:template>
</xsl:stylesheet>
```

En el ejemplo anterior, se usó un constructor `System.DateTime` (`new(2008, 4, 29)`) para crear un objeto .NET de tipo `System.DateTime`. Este objeto se creó dos veces, una vez como valor de la variable `releasedate`, y otra vez como primer y único argumento del método `System.DateTime.ToString()`. Al método de instancia `System.DateTime.ToString()` se le llama dos veces, ambas con el constructor `System.DateTime` (`new(2008, 4, 29)`) como primer y único argumento. En una de estas instancias, se usó la variable `releasedate` para obtener el objeto .NET.

Métodos de instancia y campos de instancia

La diferencia entre un método de instancia y un campo de instancia es solo teórica. En un método de instancia, se pasa directamente un objeto .NET como argumento. En un campo de instancia, se pasa un parámetro o una variable (aunque el parámetro o la variable puede contener un objeto .NET). Por ejemplo, en el código del ejemplo anterior, la variable `releasedate` contiene un objeto .NET y esta es la variable que se pasa como argumento de `ToString()` en el segundo constructor de elemento `date`. Por tanto, la instancia `ToString()` del primer elemento `date` es un método de instancia, mientras que la segunda se considera un campo de instancia. El resultado es el mismo en ambos casos.

12.2.2.2.2.4 Tipos de datos: Conversión de XPath/XQuery en .NET

Cuando se usa una función de extensión .NET dentro de una expresión XPath/XQuery, los tipos de datos de los argumentos de la función son importantes para determinar a cuál de los métodos .NET que tienen el mismo nombre se está llamando.

En .NET se siguen estas normas:

- Si en una clase hay varios métodos que tienen el mismo nombre, solamente se pueden seleccionar los métodos que tienen el mismo número de argumentos que la llamada a función.
- Los tipos de datos de cadena, numéricos y booleanos XPath/XQuery (*ver lista más abajo*) se convierten de forma implícita en el tipo de datos .NET correspondiente. Si el tipo XPath/XQuery suministrado se puede convertir en más de un tipo .NET (p. ej. `xs:integer`), entonces se selecciona el tipo .NET que se declaró para el método seleccionado. Por ejemplo, si el método .NET al que se está llamando es `fx(double)` y el tipo de datos XPath/XQuery suministrado es `xs:integer`, entonces se convierte `xs:integer` en el tipo de datos .NET `double`.

La tabla que aparece a continuación enumera las conversiones implícitas de los tipos de cadena, numéricos y booleanos XPath/XQuery en tipos de datos .NET.

`xs:string`	`StringValue, string`
`xs:boolean`	`BooleanValue, bool`
`xs:integer`	`IntegerValue, decimal, long, integer, short, byte, double, float`

`xs:float`	`FloatValue, float, double`
`xs:double`	`DoubleValue, double`
`xs:decimal`	`DecimalValue, decimal, double, float`

Los subtipos de los tipos de datos XML Schema de la tabla anterior (que se usan en XPath y XQuery) también se convierten en los tipos .NET correspondientes al tipo antecesor del subtipo.

En algunos casos quizás no sea posible seleccionar el método .NET correcto usando la información dada. Por ejemplo, imagine que:

- El argumento presentado es un valor `xs:untypedAtomic` de 10 y está destinado al método `mimétodo(float)`.
- Sin embargo, hay otro método en la clase que toma un argumento de otro tipo de datos: `mimétodo(double)`.
- Puesto que los métodos tienen el mismo nombre y el tipo suministrado (`xs:untypedAtomic`) se puede convertir correctamente tanto en `float` como en `double`, es posible que `xs:untypedAtomic` se convierta en `double` en lugar de en `float`.
- 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.

12.2.2.2.2.5 Tipos de datos: Conversión de .NET en XPath/XQuery

Cuando un método .NET devuelve un valor y el tipo de datos del valor es un tipo de cadena, numérico o booleano, entonces se convierte en el tipo de datos XPath/XQuery correspondiente. Por ejemplo, el tipo de datos .NET `decimal` se convierte en `xsd:decimal`.

Cuando se devuelve un objeto .NET o un tipo de datos que no es de cadena, numérico ni booleano, puede garantizar la conversión del tipo XPath/XQuery necesario usando primero un método .NET (p. ej. `System.DateTime.ToString()`) para convertir el objeto .NET en una cadena. En XPath/XQuery la cadena se puede modificar para ajustarse a la representación léxica del tipo necesario y convertirse después en dicho tipo (usando la expresión `cast as`, por ejemplo).

12.2.2.2.3 Scripts MSXSL para XSLT

El elemento `<msxsl:script>` contiene funciones y variables definidas por el usuario a las que se puede llamar desde dentro de expresiones XPath en la hoja de estilos XSLT. El elemento `<msxsl:script>` es un elemento de nivel superior, es decir, debe ser un elemento secundario de `<xsl:stylesheet>` o `<xsl:transform>`.

El elemento `<msxsl:script>` debe estar en el espacio de nombres `urn:schemas-microsoft-com:xslt` (*ver ejemplo más abajo*).

Lenguaje de scripting y espacio de nombres

El lenguaje de scripting utilizado dentro del bloque se especifica en el atributo `language` del elemento `<msxsl:script>` y el espacio de nombres que se debe usar para las llamadas a función desde expresiones XPath se identifica con el atributo `implements-prefix`:

```
<msxsl:script language="lenguaje-de-scripting" implements-prefix="prefijo-espacioNombres-
usuario">

   función-1 o variable-1
   ...
   función-n o variable-n

</msxsl:script>
```

El elemento `<msxsl:script>` interactúa con Windows Scripting Runtime, de modo que dentro del elemento `<msxsl:script>` solamente se pueden usar lenguajes que estén instalados en el equipo. **Para poder usar scripts MSXSL es necesario tener instalada la plataforma .NET Framework 2.0 (o superior).** Por tanto, los lenguajes de scripting .NET se pueden usar dentro del elemento `<msxsl:script>`.

El atributo `language` admite los mismos valores que el atributo `language` del elemento HTML `<script>`. Si no se especifica el atributo `language`, entonce se asume Microsoft JScript por defecto.

El atributo `implements-prefix` toma un valor que es un prefijo de un espacio de nombres declarado dentro del ámbito. Este espacio de nombres suele ser un espacio de nombres de usuario que se reservó para una biblioteca de funciones. Todas las funciones y variables definidas dentro del elemento `<msxsl:script>` están en el espacio de nombres identificado por el prefijo indicado en el atributo `implements-prefix`. Cuando se llama a una función desde dentro de una expresión XPath, el nombre de función completo debe estar en el mismo espacio de nombres que la definición de función.

Ejemplo

Aquí puede ver un ejemplo de una hoja de estilos XSLT que usa una función definida dentro de un elemento `<msxsl:script>`.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:fn="http://www.w3.org/2005/xpath-functions"
   xmlns:msxsl="urn:schemas-microsoft-com:xslt"
   xmlns:user="http://mycompany.com/mynamespace">

  <msxsl:script language="VBScript" implements-prefix="user">
    <![CDATA[
    ' Input: A currency value: the wholesale price
    ' Returns: The retail price: the input value plus 20% margin,
    ' rounded to the nearest cent
    dim a as integer  = 13
    Function AddMargin(WholesalePrice) as integer
      AddMargin = WholesalePrice * 1.2 + a
    End Function
```

```
  ]]>
  </msxsl:script>

  <xsl:template match="/">
    <html>
      <body>
        <p>
          <b>Total Retail Price =
            $<xsl:value-of select="user:AddMargin(50)"/>
          </b>
          <br/>
          <b>Total Wholesale Price =
            $<xsl:value-of select="50"/>
          </b>
        </p>
      </body>
    </html>
  </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.

12.3 Datos técnicos

Esta sección incluye información sobre algunos de aspectos técnicos de su software. La información está organizada en varios apartados:

12.3.1 Requisitos de SO y memoria

Sistema operativo

Las aplicaciones de software de Altova están disponibles en estas plataformas:

- Windows 10, Windows 11
- Windows Server 2016 o superior

Memoria

Puesto que el software está escrito en C++ no necesita tanto espacio como un JRE y suele necesitar menos memoria que otras aplicaciones similares basadas en Java. No obstante, todos los documentos se cargan en memoria por completo, para poder analizarlos completamente y mejorar la velocidad de visualización y edición. Los requisitos de memoria, por consiguiente, aumentan en función del tamaño del documento.

Los requisitos de memoria también vienen dados por el historial de operaciones Deshacer. Cuando se cortan y pegan secciones grandes de documentos de gran tamaño, la memoria disponible se puede agotar rápidamente.

12.3.2 Motores XSLT y XQuery de Altova

Validador XML

Al abrir un documento XML, la aplicación usa su validador XML integrado para comprobar si el formato es correcto, para validar el documento en relación a un esquema (si se ha especificado uno) y para generar estructuras y conjuntos de información (infosets). El validador XML también se usa para proporcionar ayuda de edición inteligente cuando usted modifique documentos y para mostrar de forma dinámica cualquier error de validación que ocurra.

El validador XML integrado implementa la recomendación final para las especificaciones 1.0 y 1.1 del esquema de XML del W3C. Altova incorpora continuamente las recomendaciones más recientes del Grupo de Trabajo del esquema de XML del W3C a su validador XML para que los productos de Altova siempre ofrezcan el entorno de desarrollo más avanzado.

Motores XSLT y XQuery

Los productos de Altova usan los motores XSLT 1.0, XSLT 2.0, XSLT 3.0, XQuery 1.0 y XQuery 3.1 de Altova. Si alguno de estos motores está incluido en el producto, encontrará documentación específica sobre el comportamiento de cada motor en la implementación en los anexos.

Nota: Altova MapForce genera código con los motores XSLT 1.0, XSLT 2.0 y XQuery 1.0.

12.3.3 Compatibilidad con Unicode

Los productos XML de Altova son completamente compatibles con Unicode. Para editar un documento XML también necesitará una fuente compatible con los caracteres Unicode utilizados por el documento.

Tenga en cuenta que la mayoría de las fuentes contienen solamente un subconjunto muy concreto de caracteres Unicode y, por tanto, están destinadas a un sistema de escritura concreto. Si algunos caracteres aparecen desfigurados, el motivo puede ser que la fuente seleccionada no contiene los glifos necesarios. Por tanto, es recomendable tener una fuente que abarque todos los caracteres Unicode. Sobre todo si edita documentos XML en varios idiomas o sistemas de escritura. Una fuente Unicode que suele venir con los equipos Windows es la fuente Arial Unicode MS.

En la carpeta `/Examples` de la carpeta de su aplicación puede encontrar un archivo XHTML llamado `UnicodeUTF-8.html` que incluye esta frase en gran número de idiomas y sistemas de escritura diferentes:

- *When the world wants to talk, it speaks Unicode*
- *Cuando el mundo quiere conversar, habla Unicode*
- *Wenn die Welt miteinander spricht, spricht sie Unicode*
- 世界的に話すなら、Unicode です。

Abra este archivo XHTML y observe el potencial de Unicode.

12.3.4 Uso de Internet

Las aplicaciones de Altova inician conexiones a Internet en estos casos:

- Si hace clic en el botón **Solicitar una clave de evaluación GRATUITA** del cuadro de diálogo "Activación del software" (**Ayuda | Activación del software**), los campos del cuadro de diálogo de activación del software se transfieren a nuestro servidor web por medio de una conexión HTTP corriente (puerto 80) y le enviamos el código de evaluación gratuito por correo electrónico.
- En algunos productos de Altova puede abrir un archivo por Internet (**Archivo | Abrir | Cambiar a URL**). En este caso, el documento se recupera usando uno de estos protocolos y conexiones: HTTP (normalmente por el puerto 80), FTP (normalmente por el puerto 20/21) o HTTPS (normalmente por el puerto 443). También puede ejecutar un servidor HTTP en el puerto 8080. (En el cuadro de diálogo "Abrir URL", después del nombre de servidor escriba dos puntos y el número de puerto.)
- Si abre un documento XML que hace referencia a un documento DTD o esquema XML y el documento se especifica a través de una URL, el documento de esquema al que se hace referencia también se recupera a través de una conexión HTTP (puerto 80) o cualquier otro protocolo (ver punto anterior). El documento de esquema también se recupera para validar el archivo XML. Recuerde que la validación

puede realizarse automáticamente nada más abrir el documento, si seleccionó esta opción en la sección *Archivo* del cuadro de diálogo "Opciones" (**Herramientas | Opciones**).

- En las aplicaciones de Altova que trabajen con WSDL y SOAP, las conexiones a servicios web son definidas por documentos WSDL.
- Si usa el comando **Archivo | Enviar por correo electrónico** de MapForce, el texto seleccionado actualmente o el archivo se envía con el programa de correo electrónico instalado en el equipo.
- Durante la activación del software y la búsqueda de actualizaciones, tal y como se describe en el contrato de licencia de software de Altova.

12.4 Información sobre licencias

En esta sección encontrará información sobre:

- la distribución de este producto de software
- la activación del software y medición de licencias
- el contrato de licencia para el usuario final que rige el uso de este producto de software

Los términos del contrato de licencia que aceptó al instalar el producto de software son vinculantes, por lo que rogamos lea atentamente toda esta información.

Para leer los términos y condiciones de cualquiera de las licencias de Altova, consulte la página de información legal de Altova en el sitio web de Altova.

12.4.1 Distribución electrónica de software

Este producto está disponible por distribución electrónica de software, un método de distribución que ofrece ventajas únicas:

- Puede evaluar el software de forma totalmente gratuita durante 30 días antes de decidir si compra el producto (*Nota: la licencia para Altova Mobile Together Designer es gratuita*).
- Si decide comprarlo, puede hacer un pedido en línea en el sitio web de Altova y conseguir en pocos minutos el software con licencia.
- Si realiza el pedido en línea, siempre recibirá la versión más reciente de nuestro software.
- El paquete de instalación del producto incluye un sistema de ayuda en pantalla al que se puede acceder desde la interfaz de la aplicación. La versión más reciente del manual del usuario está disponible en www.altova.com (i) en formato HTML y (ii) en formato PDF para descargar e imprimir si lo desea.

Período de evaluación de 30 días

Después de descargar el producto de software, puede probarlo de forma totalmente gratuita durante un plazo de 30 días. Pasados unos 20 días, el software empieza a recordarle que no tiene una licencia. El mensaje de aviso aparece una sola vez cada vez que se inicie la aplicación. Para seguir utilizando el programa una vez pasado el plazo de 30 días, deberá comprar una licencia permanente, que se entrega en forma de código clave. Para desbloquear el producto debe introducir ese código clave en el cuadro de diálogo "Activación del software".

Las licencias de los productos pueden comprarse directamente en la tienda en línea del sitio web de Altova.

Distribuir la versión de evaluación a otros usuarios de su organización

Si desea distribuir la versión de evaluación en la red de su compañía o si desea usarlo en un PC que no está conectado a Internet, solamente puede distribuir los programas de instalación (siempre y cuando no se modifiquen de forma alguna). Todo usuario que acceda al instalador debe solicitar su propio código clave de

evaluación (de 30 días). Una vez pasado este plazo de 30 días, todos los usuarios deben comprar también una licencia para poder seguir usando el producto.

12.4.2 Activación del software y medición de licencias

Durante el proceso de activación del software de Altova, puede que la aplicación utilice su red interna y su conexión a Internet para transmitir datos relacionados con la licencia durante la instalación, registro, uso o actualización del software a un servidor de licencias operado por Altova y para validar la autenticidad de los datos relacionados con la licencia y proteger a Altova de un uso ilegítimo del software y mejorar el servicio a los clientes. La activación es posible gracias al intercambio de datos de la licencia (como el sistema operativo, la dirección IP, la fecha y hora, la versión del software, el nombre del equipo, etc.) entre su equipo y el servidor de licencias de Altova.

Su producto incluye un módulo integrado de medición de licencias que le ayudará a evitar infracciones del contrato de licencia para el usuario final. Puede comprar una licencia de un solo usuario o de varios usuarios para el producto de software y el módulo de medición de licencias se asegura de que no se utiliza un número de licencias mayor al permitido.

Esta tecnología de medición de licencias usa su red de área local (LAN) para comunicarse con las instancias de la aplicación que se ejecutan en equipos diferentes.

Licencia de un solo usuario

Cuando se inicia la aplicación, se inicia el proceso de medición de licencias y el software envía un breve datagrama de multidifusión para averiguar si hay otras instancias del producto activas en otros equipos del mismo segmento de red al mismo tiempo. Si no recibe ninguna respuesta, la aplicación abre un puerto para escuchar a otras instancias de la aplicación.

Licencia de varios usuarios

Si se usa más de una instancia de la aplicación dentro de la misma red LAN, estas instancias se comunicarán entre ellas al iniciarse. Estas instancias intercambian códigos claves para que ayudarle a no sobrepasar por error el número máximo de licencias concurrentes. Se trata de la misma tecnología de medición de licencias que suele utilizarse en Unix y en otras herramientas de desarrollo de bases de datos. Gracias a ella puede comprar licencias de varios usuarios de uso concurrente a un precio razonable.

Las aplicaciones se diseñaron de tal modo que envían pocos paquetes pequeños de red y no cargan demasiado su red. Los puertos TCP/IP (2799) utilizados por su producto de Altova están registrados oficialmente en la IANA (*para más información consulte el sitio web de la IANA www.iana.org*) y nuestro módulo de medición de licencias es una tecnología probada y eficaz.

Si usa un servidor de seguridad, puede notar las comunicaciones del puerto 2799 entre los equipos que ejecutan los productos de Altova. Si quiere, puede bloquear ese tráfico, siempre y cuando esto no resulte en una infracción del contrato de licencia.

Nota sobre los certificados

Su aplicación de Altova contacta con el servidor de licencias de Altova (`link.altova.com`) vía HTTPS. Para esta comunicación, Altova usa un certificado SSL registrado. Si se reemplaza este certificado (por ejemplo, si lo reemplaza su departamento de informática o un organismo externo), entonces su aplicación de Altova le advertirá de que la conexión puede no ser segura. Si usa el certificado sustitutivo para iniciar la aplicación, lo hace por su cuenta y riesgo. Si ve un mensaje de advertencia de que la conexión puede no ser segura, compruebe el origen del certificado y consulte con su equipo técnico (que decidirán si se debe continuar con el reemplazo del certificado de Altova).

Si su organización necesita usar su propio certificado (por ejemplo, para monitorizar la comunicación hacia y desde equipos cliente), entonces recomendamos que instale en su red Altova LicenseServer, el software gratuito de gestión de licencias de Altova. Así, sus equipos cliente pueden seguir usando los certificados de su organización y AltovaLicenseServer puede usar el certificado de Altova cuando necesite comunicarse con Altova.

12.4.3 Contrato de licencia para el usuario final

- Encontrará el Contrato de licencia de Altova para el usuario final (en inglés) en: https://www.altova.com/es/legal/eula
- Encontrará la Política de privacidad de Altova en: https://www.altova.com/es/privacy

Índice

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z