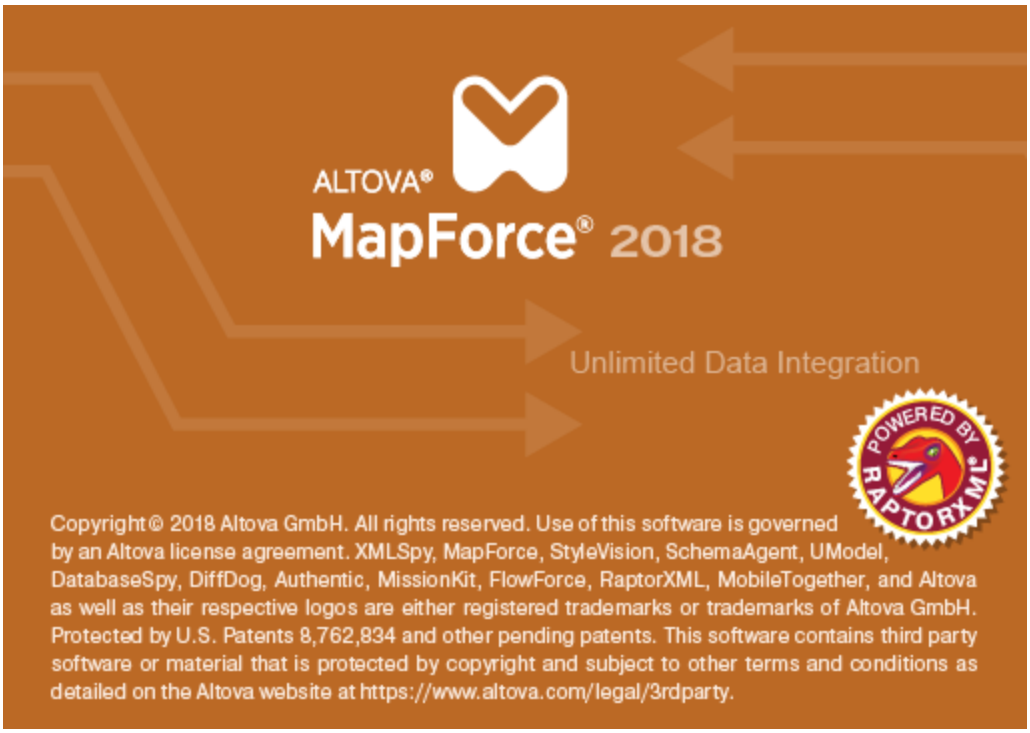
Manual del usuario y referencia



Manual del usuario y referencia de Altova MapForce 2018 Basic Edition

Fecha de publicación: 2018

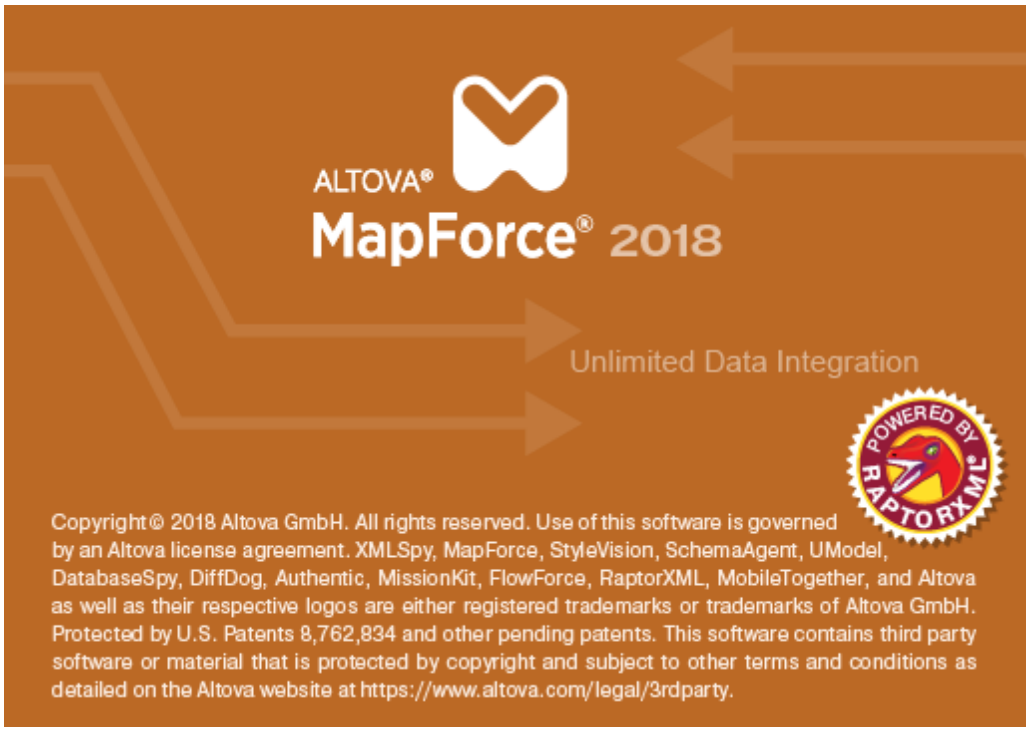
Contenido

Altova MapForce 2018 Basic Edition

Altova MapForce 2018 Basic Edition

1 Altova MapForce 2018 Basic Edition

MapForce® 2018 Basic Edition es una herramienta de asignación de datos visual para crear proyectos avanzados de integración de datos. MapForce® es una aplicación para Windows de 32/64 bits compatible con Windows Server 2008 R2 SP1 con actualización de la plataforma o superior y Windows 7 SP1 con actualización de la plataforma, Windows 8, Windows 10. Las ediciones Enterprise y Professional están disponibles en 32 y 64 bits.



Última actualización: 6/22/2018

1.1 Novedades

Novedades de **MapForce 2018**:

- Actualizaciones internas y mejoras.

Novedades de **MapForce 2017 Release 3**:

- Mejoras en las opciones de búsqueda de texto en los paneles *Resultados* y *XSLT* (*véase Búsquedas en la vista Texto*). Además, ahora la aplicación resalta texto en todos estos paneles (*véase Resaltado de texto*).
- Actualizaciones internas y mejoras.

Novedades de **MapForce 2017**:

- Ya se pueden leer nombres de nodo desde un XML de origen y asignar esta información al componente de destino. También se puede crear de forma dinámica nuevos atributos o elementos XML en el componente de destino en función de los valores que vienen del origen (*véase Asignar nombres de nodos*).
- Se pueden crear archivos de instancia XML con espacios de nombres personalizados, a nivel de elemento (*véase Declarar espacios de nombres personalizados*).
- Actualizaciones internas y mejoras.

Novedades destacadas de **MapForce 2016 R2**:

- Mejoras en el plegamiento de código en el panel *XSLT*: el texto contraído aparece con un símbolo de puntos suspensivos y tiene una vista previa en un cuadro emergente.
- Ahora puede buscar todas las instancias de una función dentro de la asignación activa (en la ventana Bibliotecas, haciendo clic con el botón derecho en la función y seleccionando el comando **Buscar todas las llamadas**).
- Actualizaciones internas y mejoras.

Novedades destacadas de **MapForce 2016**:

- Mejoras en la generación de código XSLT 1.0 (las hojas de estilos generadas ahora se pueden leer más fácilmente y se pueden ejecutar más rápido)
- Nuevas funciones de agregado en la biblioteca de MapForce: `min-string` y `max-string`. Con estas funciones puede obtener el valor mínimo y máximo de una secuencia de cadenas.

Novedades destacadas de **MapForce 2015 Release 4**:

- Actualizaciones internas y mejoras.

Novedades destacadas de **MapForce 2015 Release 3**:

- Opción para suprimir la declaración `<?xml ... ?>` en documentos XML de salida.
- Nuevo componente: Resultado simple.
- Actualizaciones internas y mejoras.

Novedades destacadas de **MapForce 2015**:

- Nuevo argumento `language` para las funciones format-date y format-dateTime.
- Nueva función de secuencia: replicate-item

Novedades destacadas de **MapForce 2014 Release 2**:

- Nuevas funciones de secuencia: generate sequence, item-at, etc.
- Opción para definir secciones CDATA en los componentes de salida.
- Opción para conservar conexiones tras la eliminación de componentes.
- Opción para resaltar automáticamente los elementos obligatorios en los componentes de destino.

Novedades destacadas de **MapForce 2014**:

- Integración del validador RaptorXML y compatibilidad básica con XML Schema 1.1
- Integración de los nuevos motores XSLT de RaptorXML
- Compatibilidad con comodines de XML Schema: xs:any y xs:anyAttribute
- Opción para usar comentarios e instrucciones de procesamiento en componentes XML de destino

Novedades destacadas de **MapForce 2013 Release 2 SP1**:

- Altova publica el rapidísimo motor de transformación

Novedades destacadas de **MapForce 2013 Release 2**:

- Configuración de final de línea definida por el usuario para los archivos de resultados.
- Actualizaciones y mejoras internas

Novedades destacadas de **MapForce 2013**:

- Actualizaciones internas y mejoras

Novedades destacadas de **MapForce 2012 Release 2**:

- Nuevo componente de Ordenación para XSLT 2.0, XQuery y el motor de ejecución integrado
- Nombres de componente definidos por el usuario

Novedades destacadas de **MapForce 2012**:

- Alineación automática de componentes en la ventana de asignación
- Avisos para conectarse al nodo primario de destino

- Reglas específicas que determinan la secuencia en la que se procesan los componentes en una asignación

Novedades destacadas de **MapForce 2011 Release 3**:

- Variables intermedias

Novedades destacadas de **MapForce 2011 Release 2**:

- Función de búsqueda en la ventana Bibliotecas
- Asignación invertida
- Función **IF-ELSE** ampliable
- Funciones **node-name** y funciones de análisis en la biblioteca de funciones **core**
-

Novedades destacadas de **MapForce 2011**:

- Vista previa de componentes intermedios en una asignación en cadena de dos o más componentes conectados a un componente de destino (vista previa de paso a través).
- Funciones de formato para tipos de datos dateTime y números para todos los lenguajes compatibles
- Mejoras en la función **auto-number**

Novedades destacadas de **MapForce 2010 Release 3**:

- Compatibilidad con valores valores nillable y con el atributo xsi:nil en archivos XML de instancia
- Posibilidad para deshabilitar la conversión automática de tipos en los documentos XML

Novedades destacadas de **MapForce 2010 Release 2**:

-
- Conexión automática de conectores secundarios idénticos al mover un conector primario
- Posibilidad de acortar cadenas de entrada antes de procesarlas

Novedades destacadas de **MapForce 2010**:

- Varios archivos de entrada/salida por componente
- Mejoras en el uso de rutas de acceso relativas
- Compatibilidad con xsi:type, permitiendo el uso de tipos derivados
- Nuevo sistema interno de tipos de datos
- Mejoras en la navegación de funciones definidas por el usuario
- Mejoras en el tratamiento de contenido mixto en elementos XML

Novedades destacadas de **MapForce 2009 SP1**:

- El usuario puede definir el orden de los parámetros en las funciones definidas por el usuario
- Posibilidad de procesar archivos XML que no son válidos con respecto al esquema XML
- Las funciones normales (estándar) definidas por el usuario ahora admiten parámetros jerárquicos complejos

Novedades destacadas de **MapForce 2009**:

- Posibilidad de usar EDI HL7 versión 3.x XML como componentes de origen y destino
- Agrupación de nodos o del contenido del nodo
- Posibilidad de filtrar los datos según la posición del nodo en una secuencia
- Compatibilidad con QName
- Búsqueda de nodos y elementos en componentes

Novedades destacadas de **MapForce 2008 Release 2**:

- Función para generar esquemas XML automáticamente a partir de archivos XML
- Función Recursos globales de Altova
- Mejoras del rendimiento

Novedades destacadas de **MapForce 2008**:

- Funciones de agregado
- Componente de búsqueda Asignación de valores
- Mejoras en las opciones de presentación de XML: pretty-print, omisión de referencias de esquema XML y configuración de codificación para cada componente
- Actualizaciones internas

Altova MapForce 2018 Basic Edition

Introducción

2 Introducción

En esta sección ofrecemos información general sobre las características de MapForce y su interfaz de usuario y definimos algunos conceptos básicos. Al final de la sección encontrará algunas notas sobre esta documentación.

2.1 Notas sobre compatibilidad

MapForce® es una aplicación Windows de 32/64 bits compatible con estos sistemas operativos:

- Windows 7 SP1 con actualización de la plataforma, Windows 8, Windows 10
- Windows Server 2008 R2 SP1 con actualización de la plataforma o superior

Solamente las ediciones Enterprise y Professional Edition están disponibles como aplicación de 64 bits.

Consulte también el apartado Datos técnicos, que incluye información general práctica sobre aspectos técnicos del software.

2.2 ¿Qué es MapForce?

Sitio web de Altova: HErramienta de asignación de datos

MapForce es un entorno IDE multiuso basado en Windows que sirve para transformar datos de un formato en otro o pasar datos de un esquema a otro mediante operaciones visuales de arrastrar y colocar. Para ello ofrece una interfaz gráfica donde el usuario no necesita crear código de programación. De hecho, MapForce genera automáticamente el código de programa necesario para realizar la transformación de datos propiamente dicha (o la asignación de datos). Si prefiere no generar código de programa, puede simplemente ejecutar la transformación con el lenguaje de transformación integrado de MapForce (disponible en las ediciones Professional y Enterprise).

Con MapForce podrá convertir y transformar datos con total comodidad, independientemente del formato. MapForce infiere automáticamente la estructura de los datos y también ofrece la posibilidad de usar esquemas para lo datos o generar el esquema automáticamente a partir de un archivo de instancia. Por ejemplo, si tiene un archivo de instancia XML pero no tiene una definición de esquema, MapForce puede generarlo automáticamente. Así podrá asignar los datos del archivo XML a otros archivos o formatos.

Estas son las tecnologías que puede usar como origen o destino de las asignaciones de datos en MapForce:

MapForce Basic Edition	MapForce Professional Edition	MapForce Enterprise Edition
• XML y XML schema • HL7 versión 3.x (basado en esquemas)	• 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, incluidas las de Microsoft Access y SQLite)	• 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 (las principales bases de datos relacionales, incluidas las de Microsoft Access y SQLite) • Familia de formatos EDI (incluidos UN/EDIFACT, ANSI X12, HL7, IATA PADIS, SAP IDoc, TRADACOMS) • Archivos JSON • Archivos Microsoft Excel 2007 y superior • Archivos de instancia y taxonomías XBRL

Estos son los lenguajes de transformación de datos disponibles en cada edición de MapForce:

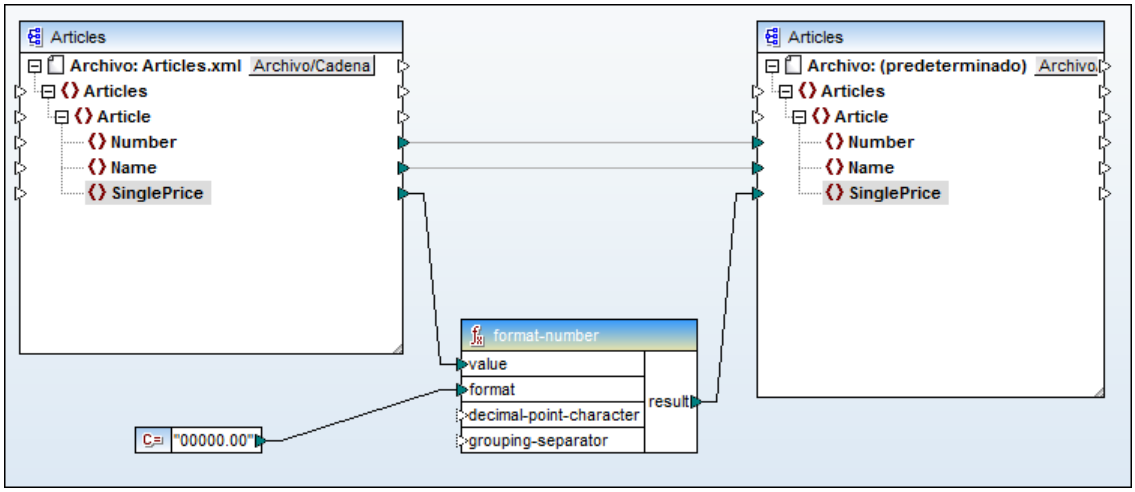
MapForce Basic Edition	MapForce Professional Edition	MapForce Enterprise Edition
• XSLT 1.0 • XSLT 2.0	• Lenguaje de transformación integrado de MapForce • XSLT 1.0 • XSLT 2.0 • XQuery • Java • C# • C++	• Lenguaje de transformación integrado de MapForce • XSLT 1.0 • XSLT 2.0 • XQuery • Java • C# • C++

En la interfaz gráfica puede consultar una vista previa de los resultados de la transformación y el código XSLT o XQuery generado. Recuerde que durante el proceso de diseño MapForce valida constantemente la integridad de los esquemas y de las transformaciones y muestra errores de validación en una ventana de mensajes. Así podrá corregir errores inmediatamente.

Cuando elija los lenguajes de transformación Java, C# o C++, MapForce generará los proyectos y las soluciones necesarios para que pueda abrirlos en Visual Studio o Eclipse directamente y

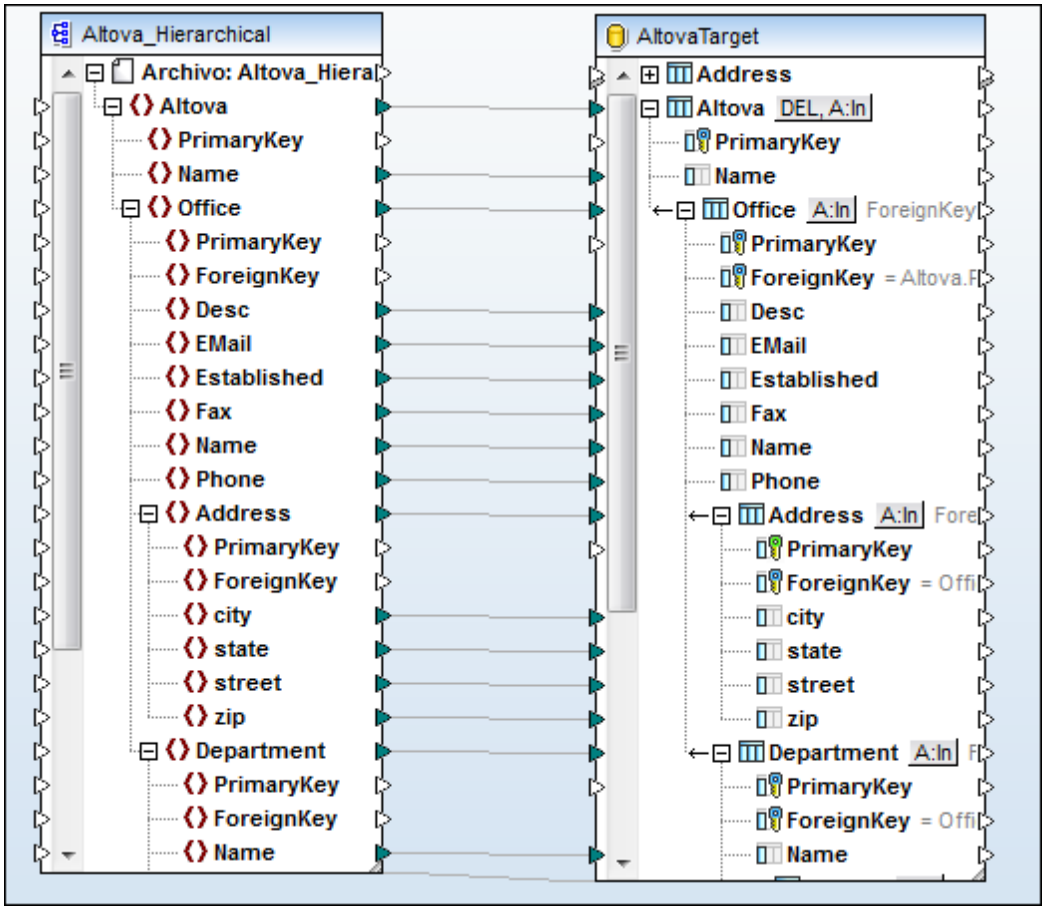
pueda ejecutar el programa de asignación de datos generado. Cuando se trate de un proyecto de integración de datos más complejo, podrá añadir código al programa generado con ayuda de las bibliotecas de Altova y de la API de MapForce.

En MapForce las transformaciones de datos se diseñan de forma visual. Por ejemplo, si trabaja con datos XML podrá conectar los elementos, atributos o comentarios de un archivo XML a los elementos, atributos o comentarios de otro archivo XML. Estas conexiones ordenan que se lean los datos del elemento o atributo de origen y se escriban en el elemento o atributo de destino.



Ejemplo de transformación de datos con dos archivos XML

Asimismo, cuando trabaje con bases de datos en la edición Professional o Enterprise, podrá ver las columnas de la BD en MapForce y crear asignaciones gráficas entre el origen y el destino. Recuerde que cuando configure la conexión a la BD, podrá elegir el controlador de BD y el tipo de conexión (ADO, ODBC o JDBC). Por último, podrá generar consultas SQL de forma gráfica, utilizar procedimientos almacenados o consultar la BD directamente (aunque las opciones disponibles dependen del tipo de BD, la edición y el controlador).



Ejemplo de transformación de datos con un archivo XML y una base de datos

La manera más sencilla de describir un diseño de asignación creado con MapForce es diciendo que el diseño lee lo datos del origen X y los escribe en el destino Y. Sin embargo, podrá crear diseños más complejos que lean datos del origen X y los escriba en Y y después lea datos del origen Y y los escriba en el destino Z. En este caso hablaríamos de asignaciones *encadenadas* o *de paso a través*. Este tipo de asignaciones permiten el acceso a datos del paso intermedio del proceso de asignación (para guardarlos en un archivo, por ejemplo).

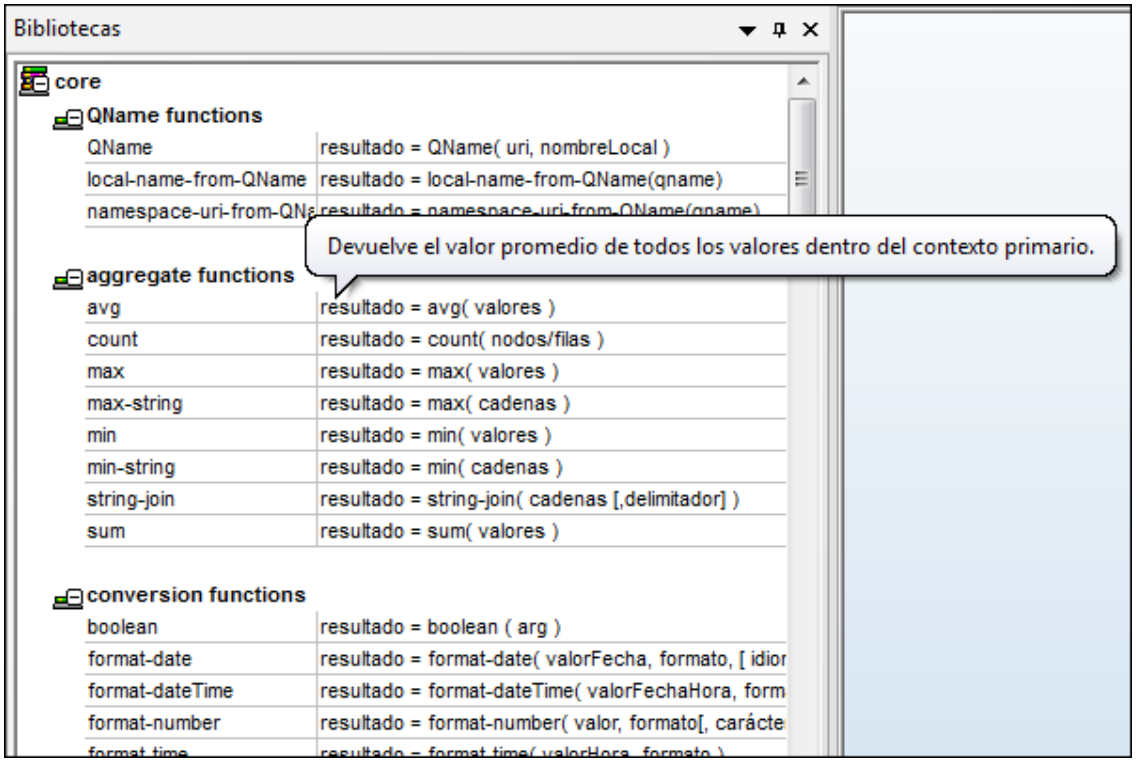
Además, en una transformación de datos de MapForce pueden procesarse varios archivos de entrada de forma dinámica y pueden generarse varios archivos de salida. Por tanto, podrá crear diseños que lean datos de varios archivos X y los escriban en un solo archivo Y o diseños que lean datos del archivo X y los escriba en varios archivos Y, etc.

Es importante tener en cuenta que se pueden mezclar varios orígenes y varios destinos en la misma transformación y que pueden ser de cualquier tipo de datos compatible. Por ejemplo, si trabaja con MapForce Professional o Enterprise, podrá combinar datos de dos bases de datos distintas en un solo archivo XML. O podrá combinar datos de varios archivos XML y escribir algunos datos en una base de datos y otros en otra. Antes de confirmar los cambios en las bases de datos podrá consultar una vista previa de las instrucciones SQL.

En ocasiones el proyecto de integración de datos exige procesar los datos (mediante ordenación, agrupación o filtrado, por ejemplo) antes de escribirlos en el destino. Por este motivo, MapForce

ofrece varios componentes funcionales muy prácticos que no son más que construcciones de lenguaje de programación simplificadas (como constantes, variables, condiciones WHERE de SQL, componentes de filtrado y ordenación, etc.). Por otra parte, MapForce ofrece extensas bibliotecas de funciones que le ayudarán a manipular todo tipo de datos.

La biblioteca integrada de funciones puede ampliarse con funciones diseñadas en MapForce (las denominadas *funciones definidas por el usuario*) o con bibliotecas externas creadas en XSLT, XQuery, Java o C#.



Ventana Bibliotecas

Si tiene demasiados archivos de asignación, puede organizarlos en un proyecto de asignación (en la edición Enterprise y Professional). Además podrá generar código de programa para el proyecto entero, en lugar de generar código para cada asignación.

Si tiene requisitos más avanzados (p. ej. cuando tenga que ejecutar transformaciones con la API de MapForce Server), puede diseñar una asignación de modo que se le puedan pasar valores en tiempo de ejecución o pueda obtener un valor de cadena simple en tiempo de ejecución. Esta característica también permite probar el resultado de funciones o asignaciones enteras que producen un valor de cadena simple. Las ediciones Enterprise y Professional también incluyen componentes que permiten analizar y serializar cadenas en tiempo de ejecución.

En la edición Enterprise puede diseñar servicios web SOAP 1.0 y SOAP 2.0 Web de forma gráfica a partir de archivos WSDL. También podrá realizar llamadas a servicios web WSDL 1.0/2.0 desde la asignación y obtener datos desde ellos. Esto incluye servicios web a los que se accede por protocolos HTTP y HTTPS y servicios web que exigen que el usuario utilice el mecanismo de seguridad de servicio web o de autenticación HTTP.

En las ediciones Enterprise y Professional puede generar documentación detallada sobre los archivos de diseño en formato HTML, Word 2007+ o RTF. El diseño de la documentación se puede personalizar (p. ej. puede incluir o excluir determinados componentes).

MapForce se integra perfectamente con otros productos de Altova MissionKit y otros productos servidor de Altova.

<table>
<tr><th>MapForce
Basic Edition</th><th>MapForce
Professional Edition</th><th>MapForce
Enterprise Edition</th></tr>
<tr><td colspan="3">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.</td></tr>
<tr><td colspan="3">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.</td></tr>
<tr><td></td><td colspan="2">Puede ejecutar las transformaciones de datos en MapForce directamente o implementarlas en otro equipo e incluso en otro sistema operativo para ejecutarlas de forma automatizada. Concretamente puede diseñar asignaciones en Windows y ejecutarlas en equipos servidor Windows, Linux o Mac que tengan instalado MapForce Server (como producto independiente o bajo control de FlowForce Server).</td></tr>
<tr><td></td><td colspan="2">Si tiene StyleVision instalado en el mismo equipo que MapForce, 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+.</td></tr>
</table>

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.

En MapForce no solo puede personalizar el aspecto del entorno de desarrollo (la interfaz gráfica del usuario), sino que además puede configurar todos los tipos de componente y de formato:

- Si se trata de una asignación de datos XML, puede elegir si se incluye una referencia de esquema o si la declaración XML se suprime en los archivos XML de salida. También puede elegir la codificación de los archivos generados (p. ej. UTF-8).
- Si se trata de una asignación de datos de BD, puede definir el tiempo de espera para la ejecución de instrucciones, puede elegir si MapForce utiliza las transacciones de la BD o si a la hora de generar código se elimina el nombre del esquema de la BD de los nombres de tabla.
- Si se trata de una asignación de datos XBRL, puede seleccionar las vistas que MapForce debe mostrar (p. ej. la vista *Bases de enlaces de presentación y definición*, la vista *Base de enlaces de tabla* o la vista *Todos los conceptos*).

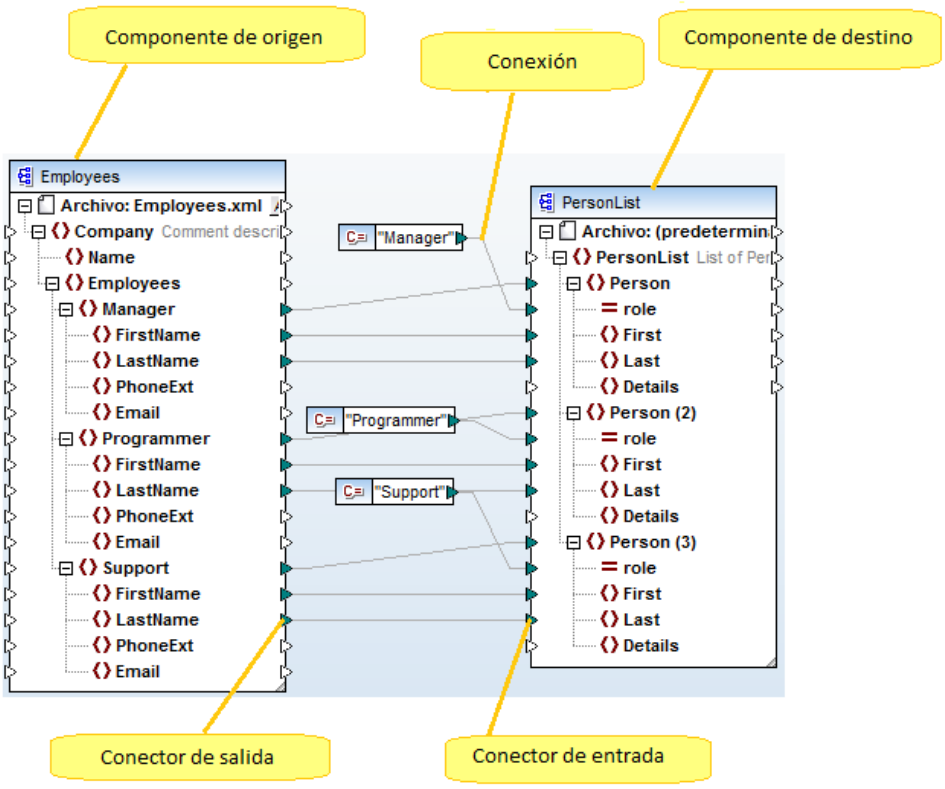
Todas las ediciones de MapForce tienen una versión de 32 bits. Además, las ediciones Enterprise y Professional están disponibles en 64 bits.

2.3 Conceptos básicos

En este apartado definimos conceptos básicos de las asignaciones de datos.

Asignación

Un diseño de asignación de datos (o simplemente *asignación de datos*) es la representación visual de cómo se deben transformar datos de un formato a otro. Una asignación está formada por componentes, que se van añadiendo al área de asignación de MapForce para crear las transformaciones de datos (p. ej. para convertir documentos XML que siguen un esquema en documentos que siguen otro esquema). 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. Los diseños de asignación de MapForce tienen la extensión de archivo **.mfd**.



Estructura básica de una asignación de datos de MapForce

Componente

En MapForce un componente es el elemento gráfico que representa visualmente la estructura (esquema) de los datos o el elemento que determina cómo se deben transformar los datos (funciones). Los componentes son piezas fundamentales necesarias para construir una asignación. En el área de asignación de MapForce los componentes aparecen en forma de rectángulo. Estos son algunos ejemplos de componente:

- Constantes
- Filtros
- Condiciones
- Funciones
- Documentos EDI (UN/EDIFACT, ANSI X12, HL7)
- Archivos Excel 2007+
- Componentes de entrada simples
- Componentes de salida simples
- Esquemas XML y documentos DTD

Conector

Un conector es un pequeño triángulo situado a la izquierda o derecha de un componente. Los conectores situados a la izquierda del componente ofrecen puntos de entrada para el componente. Los situados a la derecha ofrecen puntos de salida desde el componente.

Conexión

Una conexión es la línea que se puede dibujar para unir dos conectores. Al dibujar la línea estamos dando a MapForce instrucciones de transformar lo datos de una forma determinada (p. ej. leer datos de un documento XML y escribirlos en otro documento XML).

Componente de origen

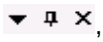
Un componente de salida es un componente donde MapForce lee datos. Al ejecutar la asignación, MapForce lee lo datos que aporta el conector del componente de origen, los convierte al tipo elegido y los envía al conector del componente de destino.

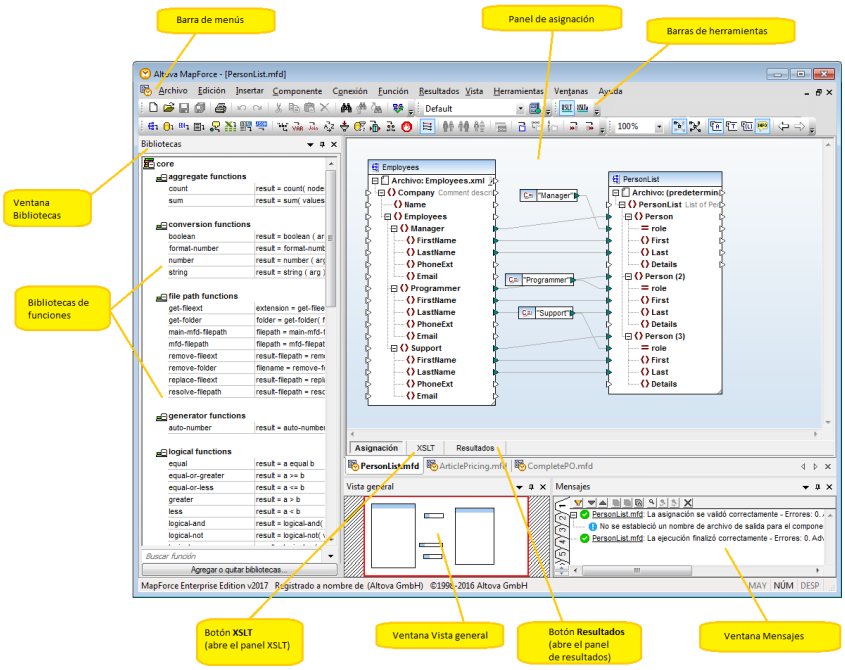
Componente de destino

Un componente de destino es un componente donde MapForce escribe datos. 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. El componente de destino es lo opuesto de un componente de origen.

2.4 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. Si necesita restaurar las barras de herramientas y ventanas a su estado predeterminado, haga clic en el comando de menú **Herramientas | Restaurar barras de herramientas y ventanas**.



Interfaz gráfica del usuario de MapForce (Basic Edition)

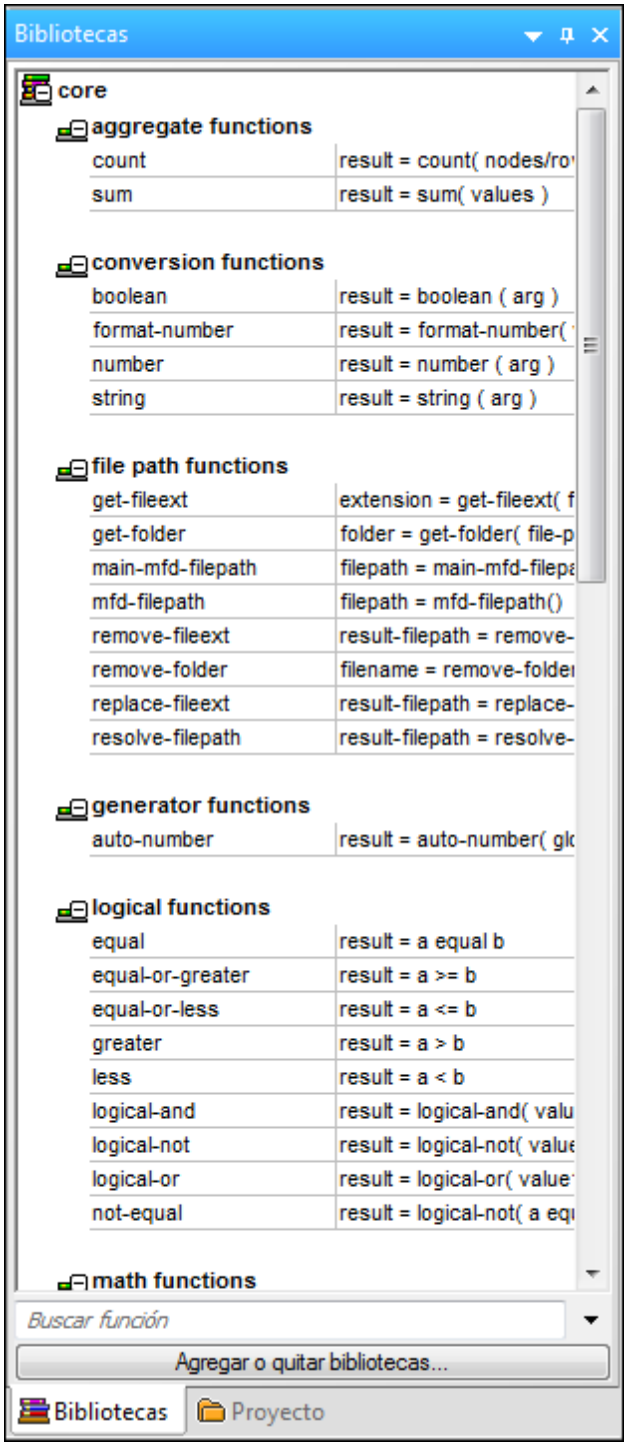
Barra de menú y barras de herramientas

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.

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. 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. Para más información consulte la sección Trabajar con funciones.

Panel *Asignación*

El panel *Asignación* es el área de trabajo donde se diseñan las asignaciones. En este panel puede añadir componentes de asignación (archivos, esquemas, constantes, variables, etc.) desde el menú **Insertar** (*véase Agregar componentes a la asignación*). Además puede arrastrar funciones desde la ventana Bibliotecas hasta el panel *Asignación* (*véase Trabajar con funciones*).

Panel *XSLT* (*XSLT2*)

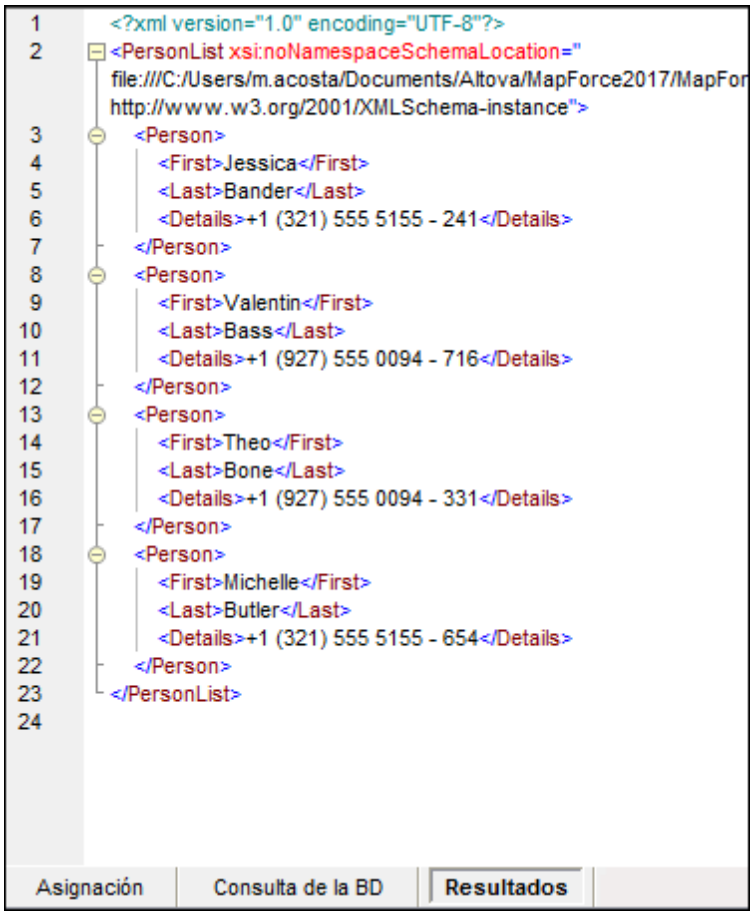
El panel *XSLT* (o *XSLT2*) muestra el código de transformación XSLT 1.0 (o 2.0) que se genera a partir de la asignación. Para cambiar a este panel seleccione el lenguaje de transformación XSLT (o XSLT 2) y después haga clic en **XSLT** (o **XSLT2**).

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 sin expandirlo, pase el cursor por encima de los puntos suspensivos y aparecerá un cuadro emergente con una vista previa del código (*imagen siguiente*). No olvide que si la vista previa no cabe en el cuadro emergente, al final del código aparecen otra vez puntos suspensivos.

Puede configurar las opciones de presentación del código (guías de sangría, marcadores de fin de línea, etc.) haciendo clic con el botón derecho dentro del panel y seleccionando el comando configurar la **Configurar la vista Texto** en el menú contextual. También puede hacer clic en el botón **Configurar la vista Texto** de la barra de herramientas.

Panel *Resultados*

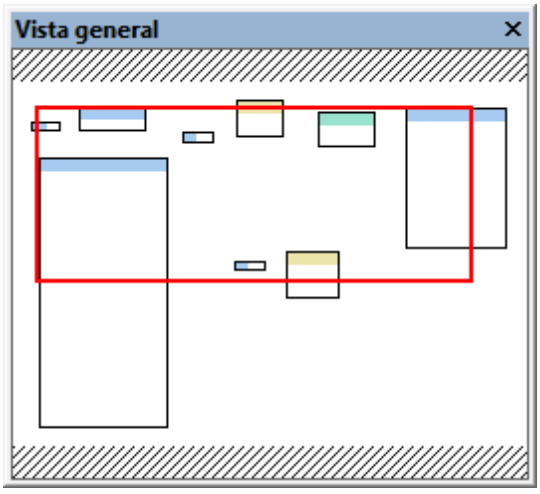
El panel *Resultados* muestra el resultado de la transformación de datos (p. ej. un archivo XML) y se abre haciendo clic en **Resultados**. Si la asignación genera varios archivos podrá navegar por echos de forma secuencial.



Este panel también incluye funciones de numeración de líneas y plegamiento de código idénticas a las del panel *XSLT*.

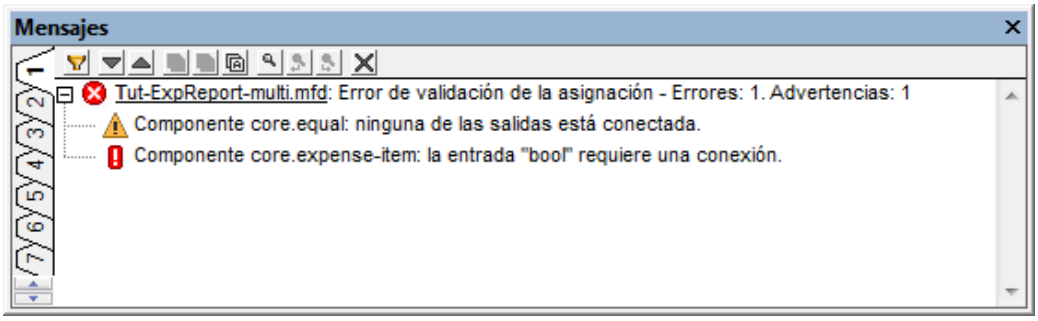
Ventana Vista general

La ventana Vista general ofrece una perspectiva general del panel *Asignación*. 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.



Ventana Mensajes

La ventana Mensajes muestra mensajes, errores y advertencias cuando se ejecuta la asignación (*véase* *Vista previa de resultados*) o cuando se valida la asignación (*véase* *Validar asignaciones*).



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.

El resultado de la operación de ejecución o validación aparece en la ventana Mensajes acompañado de uno de estos iconos de estado:

Icono	Descripción
	La operación finalizó correctamente.
	La operación finalizó con advertencias.
	Error en la operación.

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

Icono	Descripción
	Mensaje de información. La ejecución de la asignación no se interrumpe.
	Mensaje de advertencia. La ejecución de la asignación no se interrumpe. Pueden generarse 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. Cuando ocurre un error falla la ejecución de la asignación y no se genera ningún resultado. Tampoco es posible acceder a la vista previa del código XSLT o XQuery.

Además de los iconos anteriores la ventana Mensajes ofrece botones para completar estas acciones:

Icono	Descripción
	Filtra los mensajes según el nivel de gravedad (información, errores, 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 siguiente línea.
	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.
	Borra 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 ejecutar o validar la asignación, seleccione la pestaña numerada correspondiente en el lateral izquierdo de la ventana Mensajes.

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 la aplicación. Por ejemplo, muestra información contextual sobre el botón o comando señalado con el puntero del ratón. 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 incluye ningún sufijo.

2.5 Notas generales

Archivos de ejemplo

La mayoría de los archivos de diseño de asignación de datos que aparecen en esta documentación pueden encontrarse en estas carpetas:

- `C:\Usuarios\<usuario>\Documentos\Altova\MapForce2018\MapForce Examples`
- `C:\Usuarios\<usuario>\Documentos\Altova\MapForce2018\MapForce Examples\Tutorials`

Las asignaciones de datos y archivos de instancia que vienen con MapForce sirven para aprender a usar la mayoría de las características del software. Por este motivo le animamos a crear copias de estos archivos y experimentar con ellas.

Interfaz gráfica del usuario

Algunas imágenes (capturas de pantalla) que acompañan a esta documentación muestran elementos de la interfaz gráfica del usuario que quizás no correspondan a la edición de MapForce que tiene instalada. Por lo general, las capturas de pantalla incluyen el nombre del diseño de asignación (archivo **`*.mfd`**), así como la edición de MapForce a la que corresponde la imagen.

Altova MapForce 2018 Basic Edition

Tutoriales

3 Tutoriales

Los tutoriales de MapForce están diseñados para ayudarle a comprender y a utilizar las funciones básicas de transformación de datos de MapForce lo más rápido posible. El objetivo de este *curso rápido sobre MapForce* no es explicar en detalle todas las funciones de MapForce, sino describir sus funciones básicas paso a paso por lo que recomendamos seguir los tutoriales en el orden correcto. También recomendamos comprender bien todos los conceptos de un tutorial antes de pasar al siguiente porque los tutoriales aumentan en complejidad. También es recomendable tener algo de experiencia previa con XML y XML Schema.

Pasar datos XML a un esquema nuevo

En este tutorial aprenderá a pasar datos de una estructura XML a otra usando el lenguaje XSLT 2.0 y sin necesidad de escribir ni una línea de código. El tutorial también define los conceptos de *secuencias* y *elementos* y explica cómo crear conexiones de asignación, usar funciones, validar una asignación y obtener una vista previa y guardar los resultados en el disco.

Asignar varios orígenes de datos a un destino

En este tutorial aprenderá a leer datos de dos archivos XML con esquemas diferentes y combinarlos en un solo archivo XML de destino. El tutorial también explica cómo se cambia el nombre y los archivos de instancia de cada componente de la asignación y define el concepto de *duplicados de entrada*.

Trabajar con varios esquemas de destino

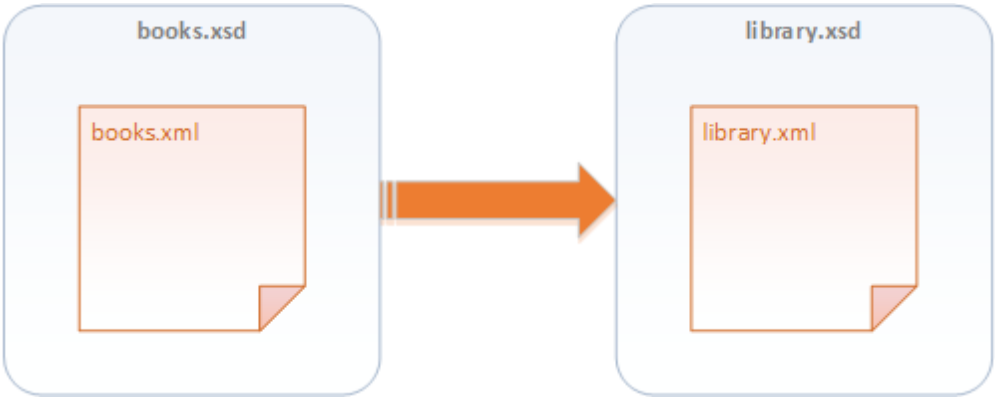
En este tutorial aprenderá a trabajar con asignaciones más complejas que producen dos o más resultados. Concretamente aprenderá a generar en la misma asignación un archivo XML que almacena una lista de libros y otro archivo XML que solo contiene un subconjunto de los libros del primer archivo, filtrados por año de publicación. Para ello deberá utilizar el componente **Filtro**, una función y una constante numérica.

Procesar y generar archivos de forma dinámica

En este tutorial aprenderá a leer datos de varios archivos XML de instancia ubicados en la misma carpeta y escribir los datos en varios archivos XML generados instantáneamente. El tutorial también explica cómo quitar las declaraciones XML y de esquema y cómo usar funciones para concatenar cadenas y extraer extensiones de archivo.

3.1 Pasar datos XML a un esquema nuevo

Este tutorial explica cómo pasar datos de un archivo XML a otro y define los conceptos básicos del entorno de desarrollo de MapForce. Ambos archivos XML sirven para almacenar una lista de libros, pero sus elementos tienen nombres diferentes y siguen estructuras distintas (es decir, los archivos tienen diferentes esquemas).



Modelo abstracto de la transformación de datos

El fragmento de código que aparece a continuación muestra algunos datos del archivo que usaremos como origen de datos (para mayor simplicidad se omitieron las declaraciones XML y de espacio de nombres).

```
<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>
```

books.xml

Y a continuación puede ver los mismos datos en el archivo de destino:

```
<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>
```

library.xml

Como puede observar, algunos nombres de elemento no coinciden en el XML de origen y de destino. Nuestro objetivo es rellenar los elementos `<author>`, `<title>`, `<genre>` y `<publish_year>` del archivo de destino con los datos de los elementos equivalentes del archivo de origen (`<author>`, `<title>`, `<category>`, `<year>`). El atributo `id` del archivo XML de origen debe asignarse al elemento `<id>` del archivo XML de destino. Por último, debemos rellenar el elemento `<last_updated>` del archivo XML de destino con la fecha y la hora en que se actualizó el archivo por última vez.

Para conseguir nuestro objetivo debemos seguir estos pasos:

Paso nº1: seleccionar XSLT2 como lenguaje de transformación

Esto puede hacerse de dos maneras distintas:

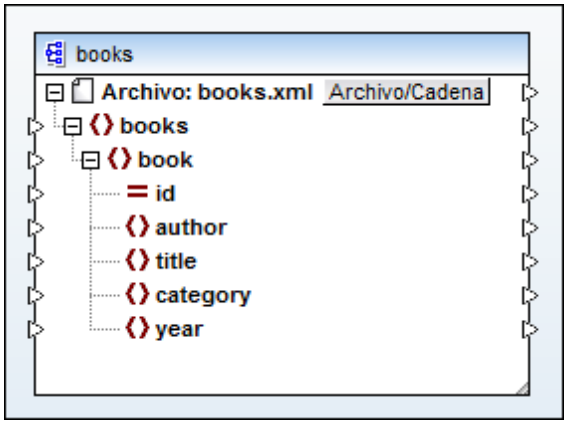
- Haciendo clic en el botón **XSLT2**  de la barra de herramientas.
- Haciendo clic en el comando **XSLT 2.0** del menú **Resultados**.

Paso nº2: agregar el archivo XML de origen a la asignación

El archivo XML de origen para la asignación está en esta ruta de acceso `<Documentos>\Altova\MapForce2018\MapForceExamples\Tutorial\books.xml`. Hay varias maneras de añadir un archivo XML de origen a la asignación:

- Haciendo clic en el botón **Insertar archivo o esquema XML**  de la barra de herramientas.
- Haciendo clic en el comando **Archivo o esquema XML** del menú **Insertar**.
- Arrastrando el archivo XML desde el explorador de Windows hasta el panel de asignación.

Una vez añadido el archivo al panel de asignación, podrá ver su estructura, definida claramente de forma gráfica. En MapForce esta estructura se denomina componente de asignación o *componente* sencillamente. Puede expandir y contraer los elementos del componente con los iconos ⊟ y ⊞ o pulsando las teclas **+** y **-** del teclado numérico.



Componente de asignación

Para mover el componente a otra posición del panel de asignación basta con arrastrarlo por la barra de título. Para ajustar su tamaño haga clic en la esquina del componente y arrastre el puntero del ratón. También puede hacer doble clic en la esquema y MapForce ajustará el tamaño del componente automáticamente.

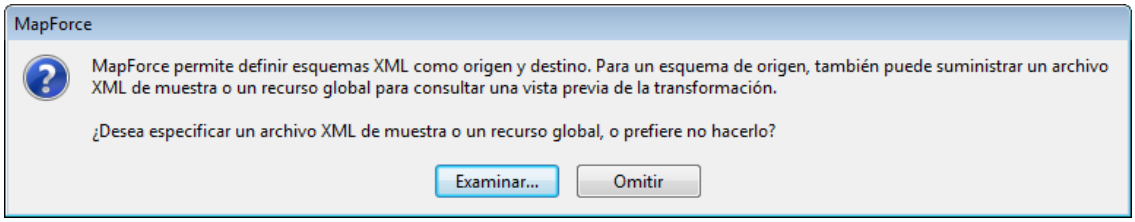
El nodo de nivel superior representa el nombre del archivo (en este caso concreto se trata del nombre del archivo XML). Los elementos XML de la estructura se representan con el icono , mientras que los atributos XML se representan con el icono .

Los pequeños triángulos que aparecen en los laterales del componente representan entradas (si están en el lado izquierdo) o salidas (si están en el lado derecho) de la asignación. En MapForce reciben el nombre de *conectores de entrada* y *conectores de salida* respectivamente.

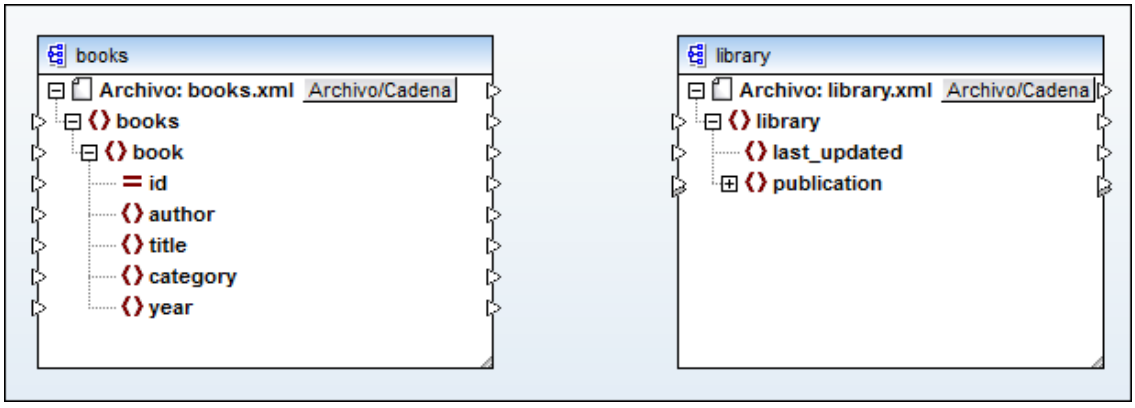
Paso nº3: agregar el esquema XML de destino a la asignación

Para generar el archivo XML de destino usaremos un esquema XML ya existente aunque también podría ser un archivo dado por un colaborador o un esquema XML nuevo creado con XMLSpy, por ejemplo. Si no tiene un esquema para los datos XML, MapForce solicita que genere uno cada vez que se añade a la asignación un archivo XML que no cuenta con la referencia de esquema correspondiente.

Para nuestro ejemplo usamos el esquema XML `<Documentos>\Altova\MapForce2018\MapForceExamples\Tutorial\library.xsd`. Para agregarlo a la asignación hacemos lo mismo que hicimos para añadir el archivo XML de origen (es decir, hacemos clic en el botón **Insertar archivo o esquema XML** de la barra de herramientas). Cuando MapForce solicite un archivo de instancia haga clic en **Omitir**.



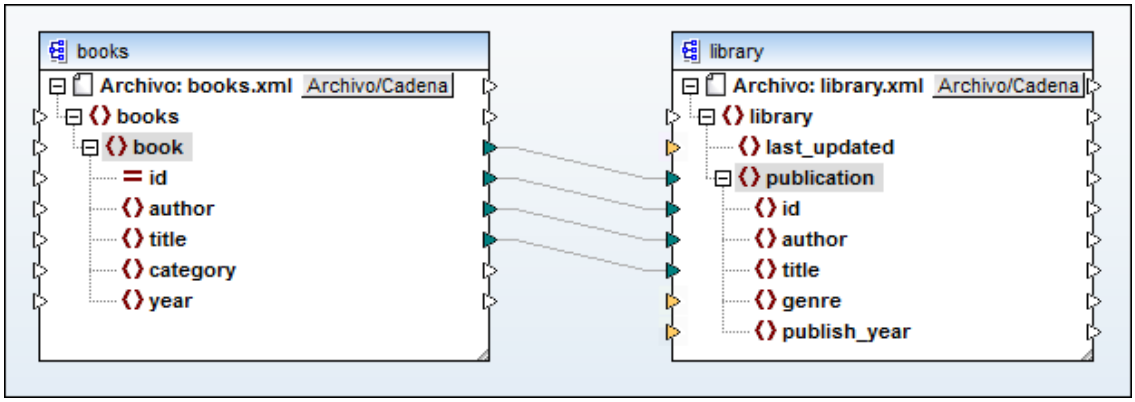
En este momento el diseño de asignación tiene este aspecto:



Paso nº4: crear las conexiones

Por cada elemento <book> del archivo XML de origen, queremos crear un elemento <publication> nuevo en el archivo XML de destino. Por tanto, crearemos una conexión de asignación entre el elemento <book> del componente de origen y el elemento <publication> del componente de destino. Para crear la conexión de asignación basta con hacer clic en el conector de salida (el triángulo situado a la derecha del elemento <book>) y arrastrarlo hasta el conector de entrada del elemento <publication> del componente de destino.

A continuación MapForce crea automáticamente una conexión entre todos los elementos secundarios de <book> y los elementos del mismo nombre del componente de destino. Es decir, MapForce crea 4 conexiones simultáneamente. Esta característica recibe el nombre de *conexión automática de secundarios equivalentes* y puede deshabilitarse o configurarse, según corresponda.



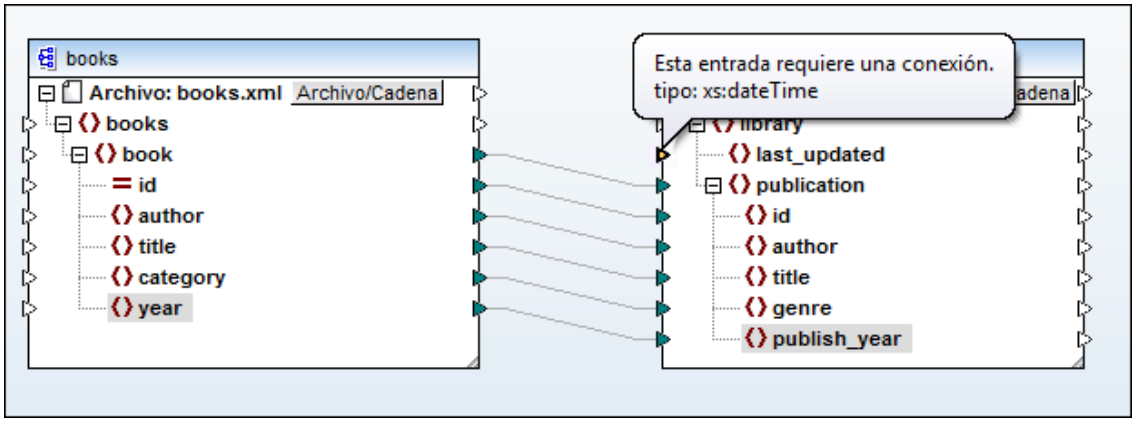
La función de conexión automática de secundarios equivalentes se puede habilitar y deshabilitar de dos formas distintas:

- Haciendo clic en el botón **Alternar la conexión automática de secundarios** de la barra de herramientas.
- Haciendo clic en el comando **Conectar automáticamente los secundarios equivalentes** del menú **Conexión**.

Observe que MapForce resaltó en color naranja algunos conectores de entrada del componente de destino. Esto significa que estos elementos son obligatorios. Para garantizar la validez del archivo XML de destino es necesario indicar valores para estos elementos obligatorios:

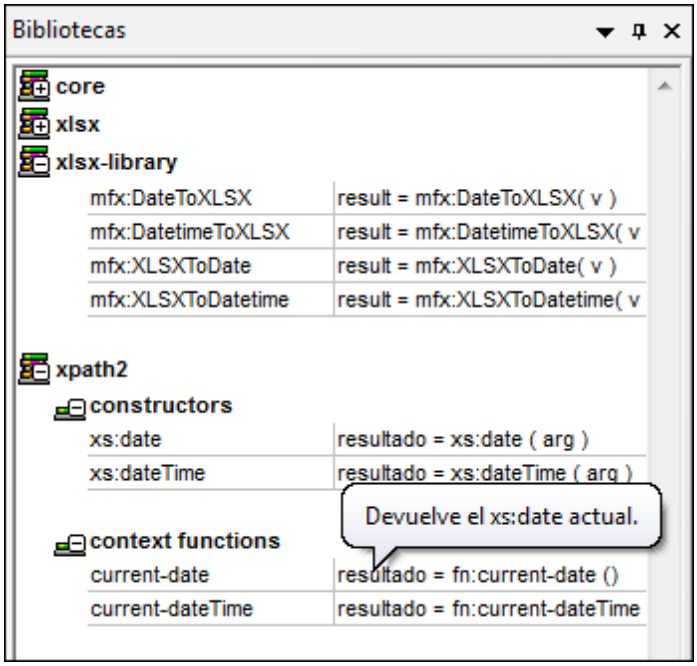
- Conectamos el elemento `<category>` del componente de origen con el elemento `<genre>` del componente de destino.
- Conectamos el elemento `<year>` del componente de origen con el elemento `<publish_year>` del componente de destino.

Por último debemos indicar un valor para el elemento `<last_updated>`. Si pasamos el puntero por encima de su conector de entrada podemos ver que el elemento es de tipo `xs:dateTime`. Recuerde que para poder ver esta información rápida debe habilitar el botón **Mostrar información rápida** de la barra de herramientas.



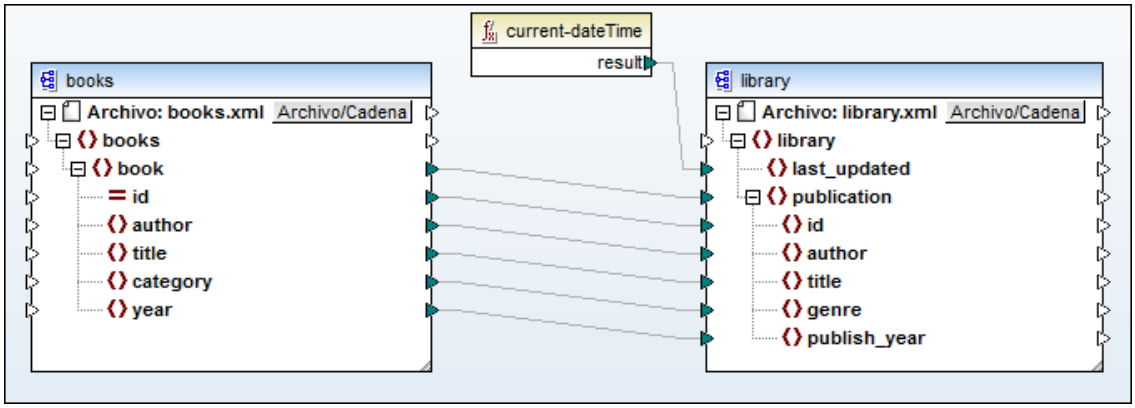
También puede cambiar la configuración para que el tipo de datos siempre esté visible (haciendo clic en el botón **Mostrar tipos de datos** de la barra de herramientas).

Para obtener la fecha y hora actuales (es decir, el valor `xs:dateTime`) podemos usar una función XSLT de fecha y hora. Para buscar esta función XSLT tecleamos `date` en el cuadro de texto situado en la parte inferior de la ventana Bibliotecas.

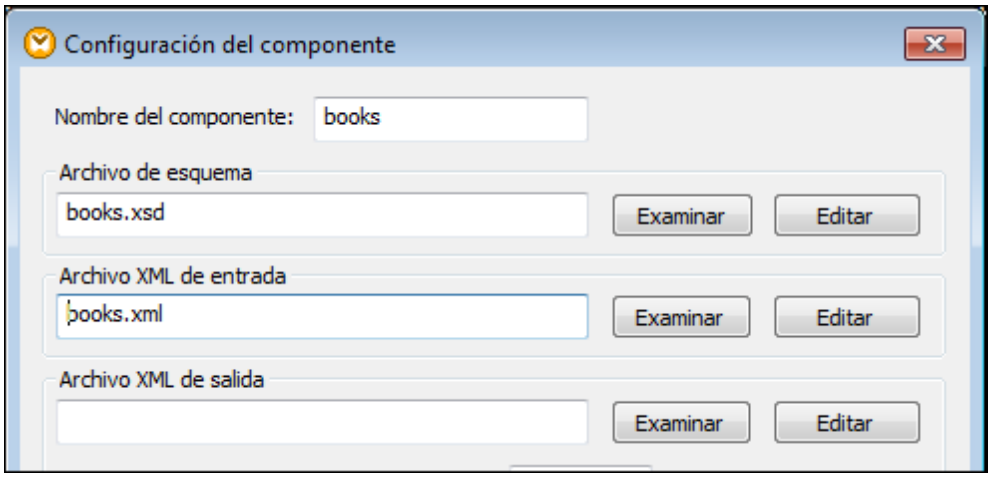


Tal y como puede ver en la imagen anterior, al pasar el puntero por la segunda columna de la función, aparece su descripción. Para que esta descripción aparezca debe estar habilitado el botón **Mostrar información rápida** de la barra de herramientas.

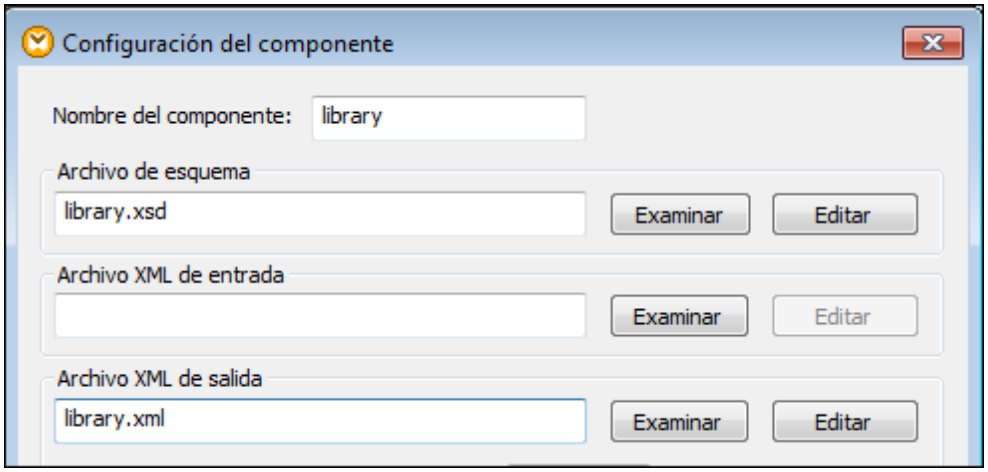
Para agregar la función a la asignación arrástrela hasta el panel de asignación y conecte su salida a la entrada del elemento <last_updated>.



El diseño de asignación ya está terminado y pasa los datos del archivo de instancia **books.xml** (que tiene el esquema **books.xsd**) a un archivo nuevo llamado **library.xml** (que tiene el esquema **library.xsd**). Si hacemos doble clic en el título de un componente, aparece el cuadro de diálogo "Configuración del componente":



Configuración del componente de origen



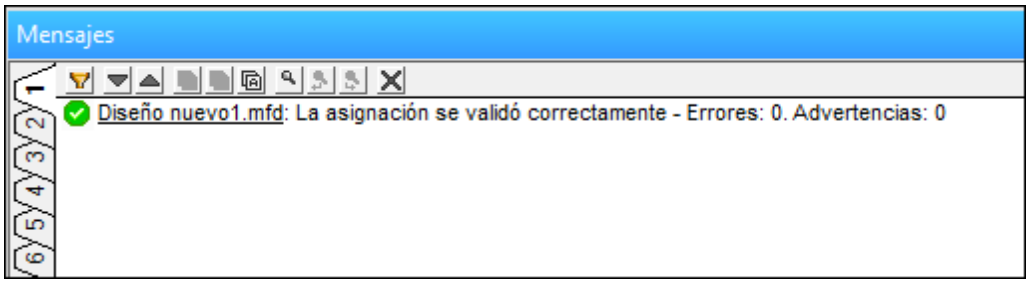
Configuración del componente de destino

Paso nº5: validar y guardar la asignación de datos

El paso de validar la asignación de datos es opcional pero importante porque permite corregir errores y ver advertencias antes de ejecutar la asignación. Hay dos maneras de comprobar si la asignación es válida:

- Haciendo clic en el comando **Validar asignación** del menú **Archivo**.
- Haciendo clic en el botón **Validar** de la barra de herramientas.

La ventana Mensajes muestra los resultados de la validación:



Ventana Mensajes

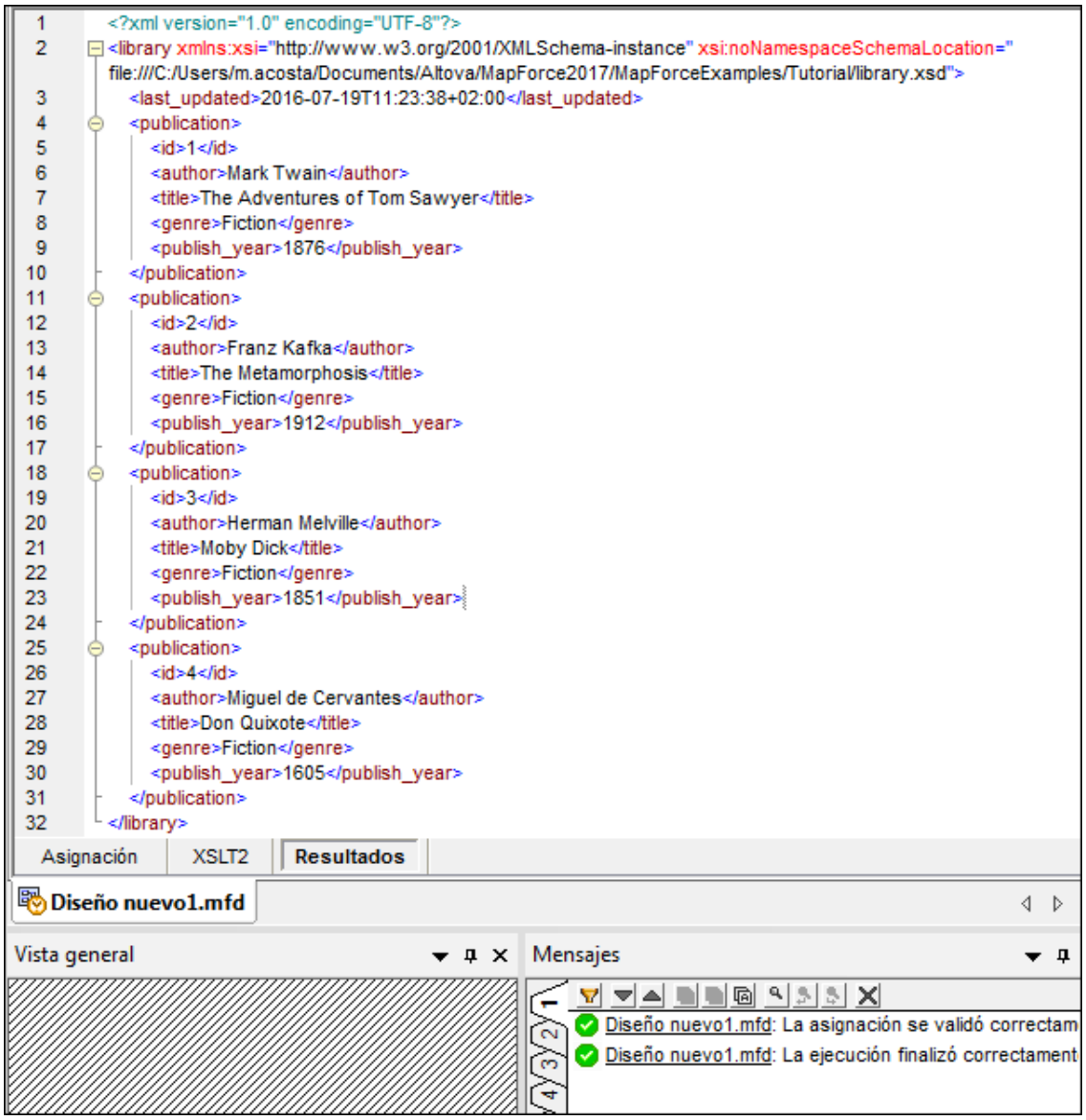
Llegados a este también puede guardar la asignación en un archivo:

- Haciendo clic en el comando Guardar del menú Archivo.
- Haciendo clic en el botón **Guardar** de la barra de herramientas.

Este diseño de asignación está en esta ruta de acceso: `<Documentos>\Altova\MapForce2018\MapForceExamples\Tutorial\\BooksToLibrary.mfd`. Es decir, a partir de ahora puede continuar trabajando con el archivo de asignación que creó siguiendo las instrucciones del tutorial o continuar con el archivo **BooksToLibrary.mfd**.

Paso nº6: vista previa del resultado de la asignación

MapForce puede ofrecer una vista previa de lo resultados de la asignación de datos. Para obtener esta vista previa basta con hacer clic en el botón **Resultados** situado en la parte inferior del panel de asignación. MapForce ejecuta la transformación de datos y muestra los resultados en el panel *Resultados*.



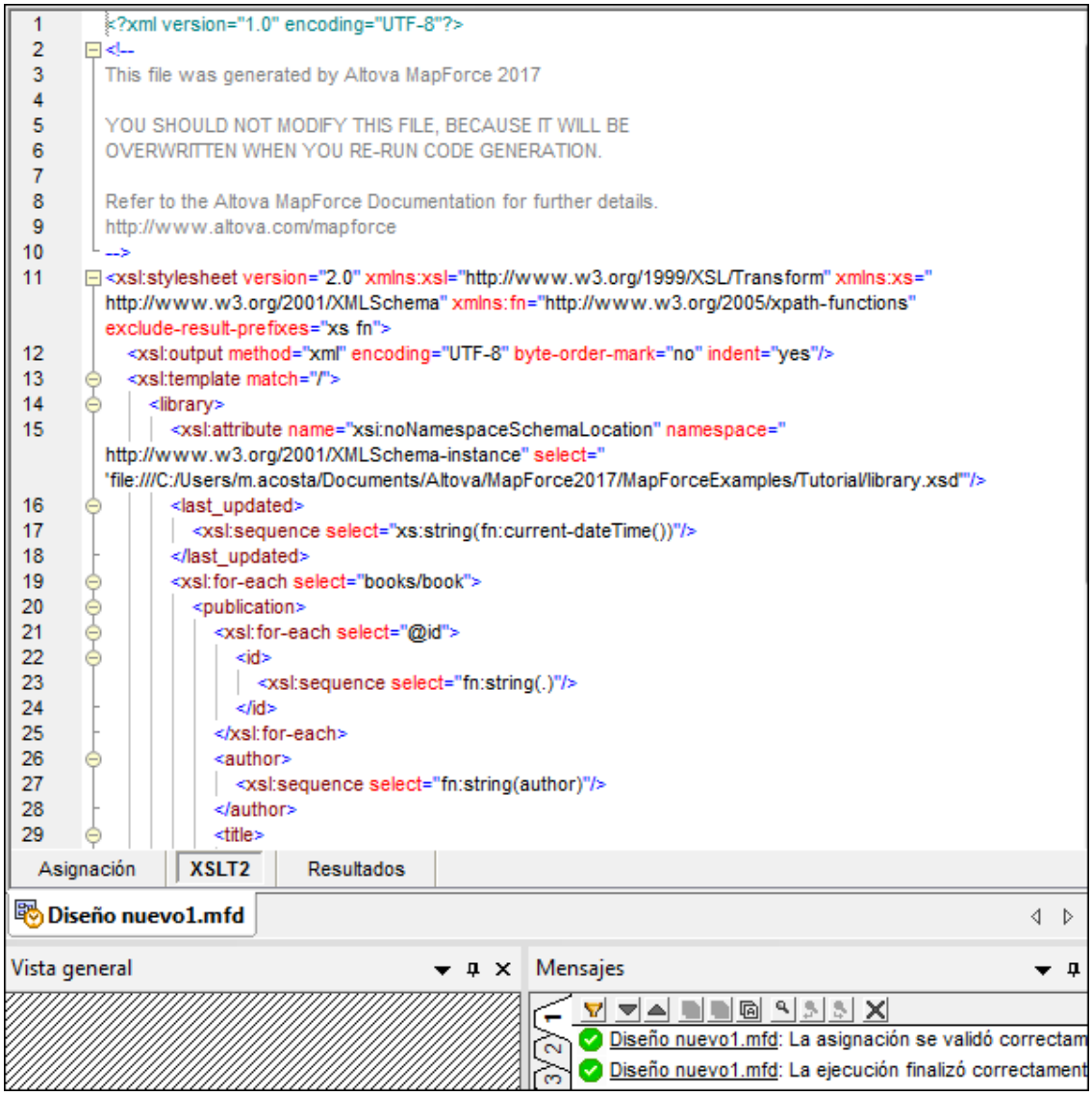
Panel Resultados

Ahora puede ver en MapForce los resultados de la transformación.

La vista previa del panel *Resultados* no se guarda en disco, sino que MapForce crea archivos de resultados temporales. Para guardar en disco el archivo que muestra el panel *Resultados*, seleccione el comando **Resultados Guardar el archivo de salida** o haga clic en el botón **Guardar resultado generado**  de la barra de herramientas.

Si quiere que MapForce escriba los resultados en archivos finales directamente, haga clic en **Herramientas | Opciones | Generales** y marque la casilla *Escribir directamente en archivos de salida finales*. Sin embargo, no recomendamos marcar esta casilla mientras realiza el tutorial para evitar sobrescribir datos en los archivos del tutorial originales.

También puede crear una vista previa del código XSLT generado que realiza la transformación. Para ello basta con hacer clic en el botón **XSLT2** situado en la parte inferior del panel de asignación.



Panel XSLT2

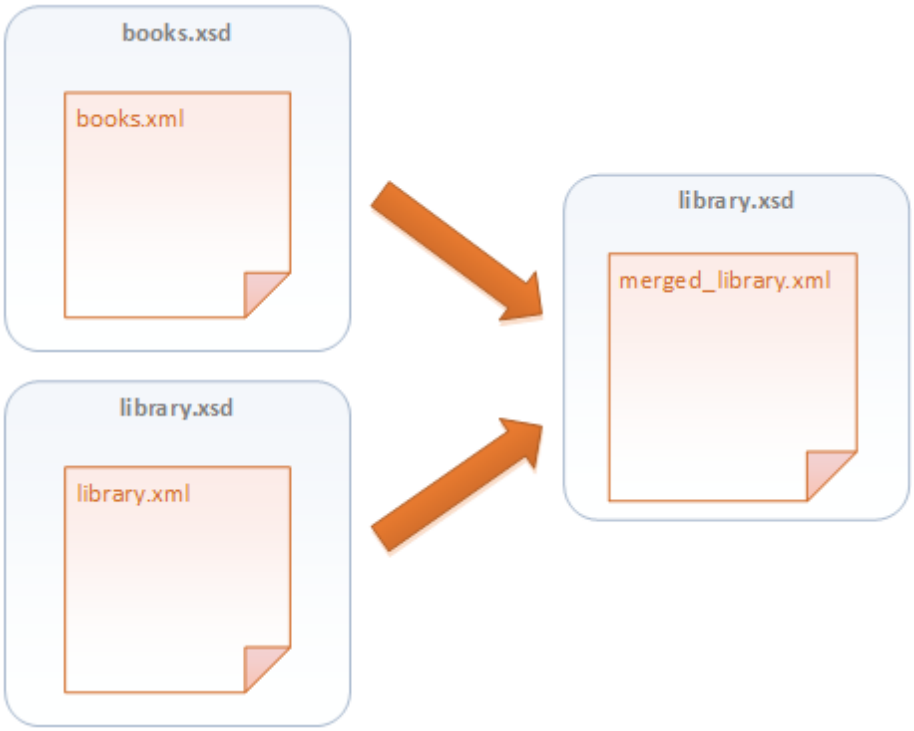
Para generar y guardar el código XSLT2 en un archivo seleccione el comando **Archivo | Generar código en | XSLT 2.0**. Cuando se le solicite, seleccione la carpeta donde se debe guardar el código generado. Una vez generado el código, la carpeta de destino incluirá estos dos archivos:

1. Un archivo de transformación XSLT que se llama como el esquema de destino (p. ej. **MappingMaptolibrary.xslt**).
2. Un archivo **DoTransform.bat**, que sirve para ejecutar la transformación XSLT con RaptorXML Server (consulte https://www.altova.com/es/raptorxml.html para obtener más información).

3.2 Asignar varios orígenes de datos a un destino

En el tutorial anterior pasamos datos de un archivo de origen (**books.xml**) a un archivo de destino (**library.xml**). El archivo de destino no existía antes de ejecutar la asignación, es decir, se generó al transformar la asignación. Imaginemos ahora que ya tuviéramos un archivo llamado **library.xml** y que deseáramos combinar sus datos con los datos convertidos de **books.xml**.

En este tutorial el objetivo es diseñar una asignación de datos que genere un archivo llamado **merged_library.xml**. El archivo de resultados generado incluirá los datos de ambos archivos **books.xml** y **library.xml**. Recuerde que los archivos utilizados como fuente (**books.xml** y **library.xml**) tienen esquemas diferentes. Si los archivos fuente tuvieran el mismo esquema, también podría combinar sus datos usando una técnica diferente (consulte el tutorial *Procesar y generar archivos de forma dinámica*).



Modelo abstracto de la transformación de datos

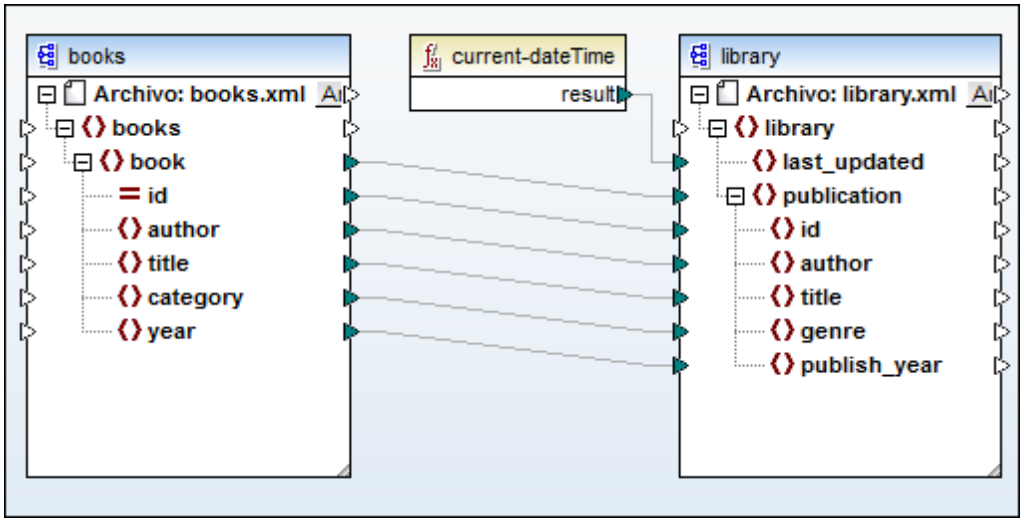
Para conseguir nuestro objetivo debemos seguir estos pasos:

Paso nº1: preparar el archivo de diseño de asignación

Para este tutorial usamos como punto de partida la asignación **BooksToLibrary.mfd** de la carpeta `<Documentos>\Altova\MapForce2018\MapForceExamples\Tutorial\`. Esta asignación es la que diseñamos en el primer tutorial (*Pasar datos XML a un esquema nuevo*). Para empezar abrimos el archivo **BooksToLibrary.mfd** en MapForce y lo guardamos con otro nombre.

Asegúrese de guardar la asignación nueva en la carpeta `<Documentos>\Altova`

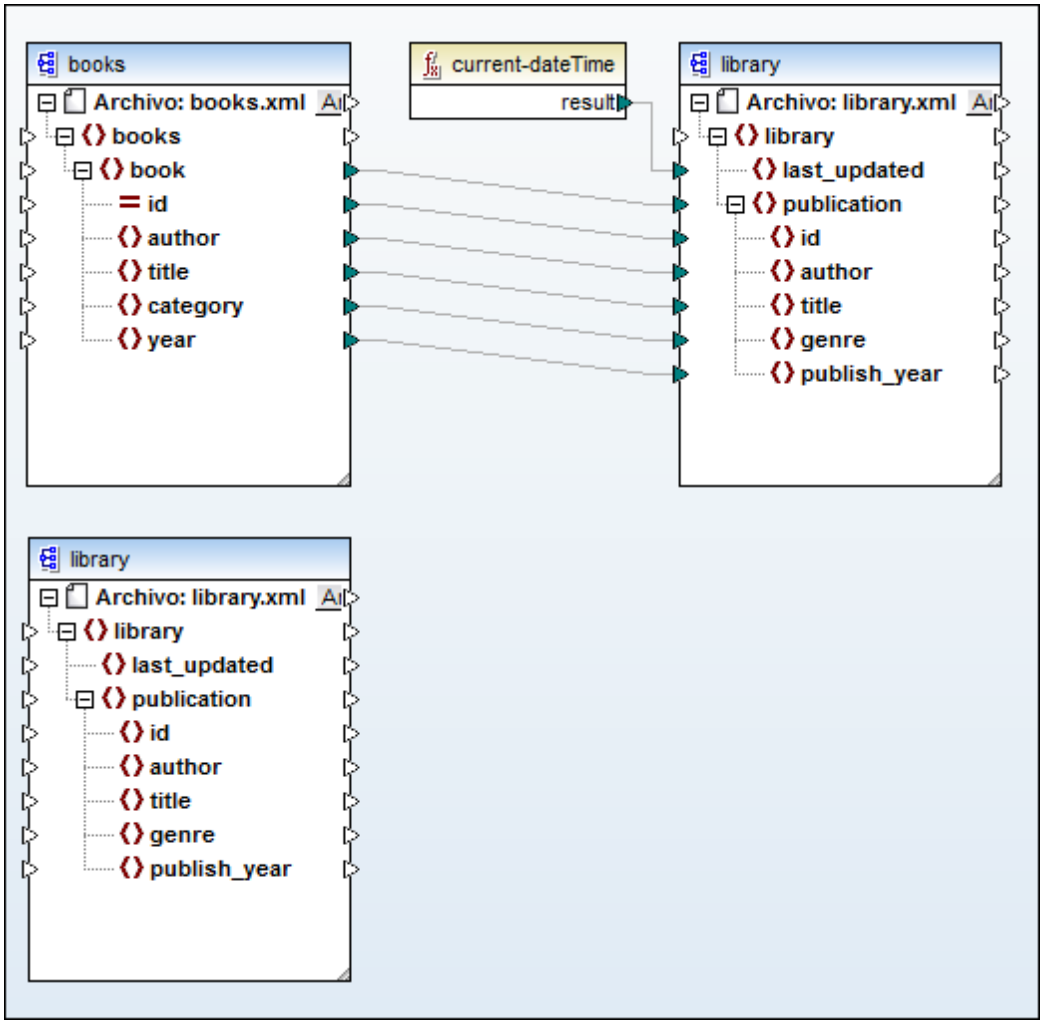
`\MapForce2018\MapForceExamples\Tutorial\` porque hace referencia a varios archivos que están en esa carpeta.



BooksToLibrary.mfd

Paso nº2: crear otro componente de origen

Primero seleccionamos el componente de destino, lo copiamos (**Ctrl+C**) y lo pegamos (**Ctrl+V**) en el panel de asignación. Haga clic en el título de este nuevo componente y colóquelo debajo del componente **books**.

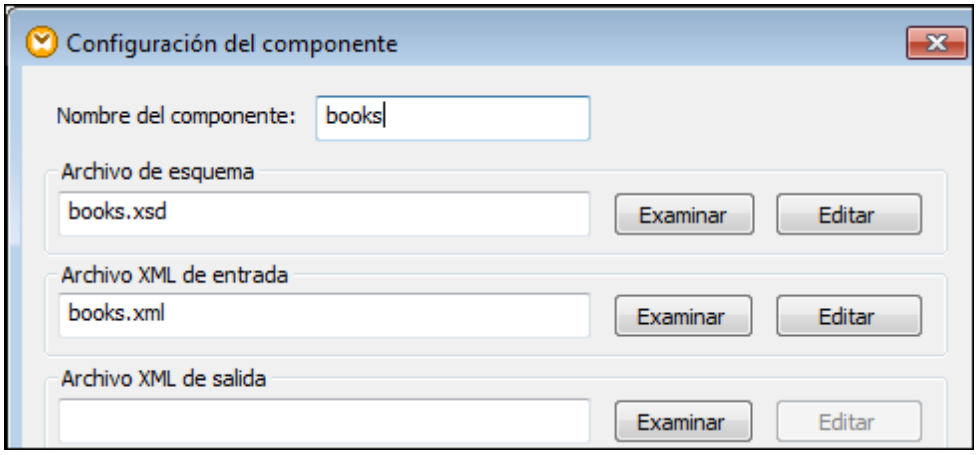


Ahora la asignación tiene dos componentes de origen (**books** y **library**) y un componente de destino (**library**).

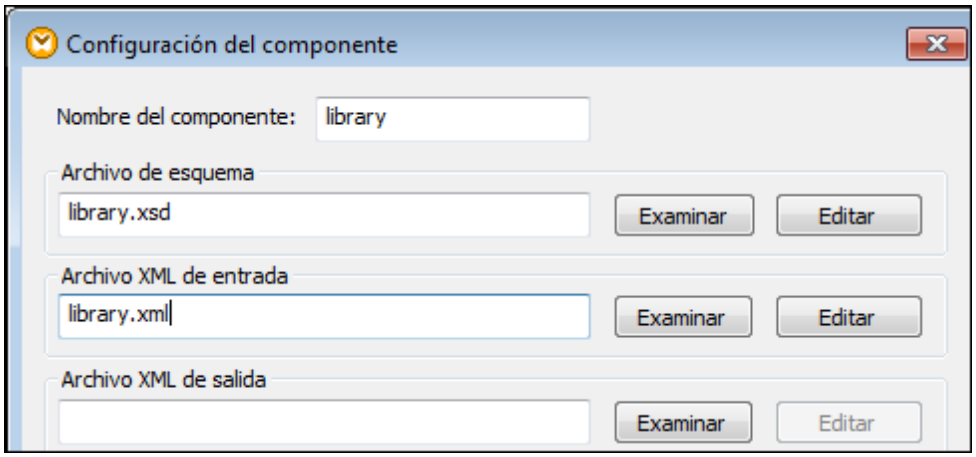
Los componentes de asignación se pueden mover en cualquier dirección (izquierda, derecha, abajo, arriba). Sin embargo, las tareas de asignación serán más fáciles si coloca los componentes de origen a la izquierda de los componentes de destino. Esta es la convención utilizada en esta documentación y en los archivos de ejemplo que vienen con MapForce.

Paso nº3: verificar y configurar los archivos de entrada y salida

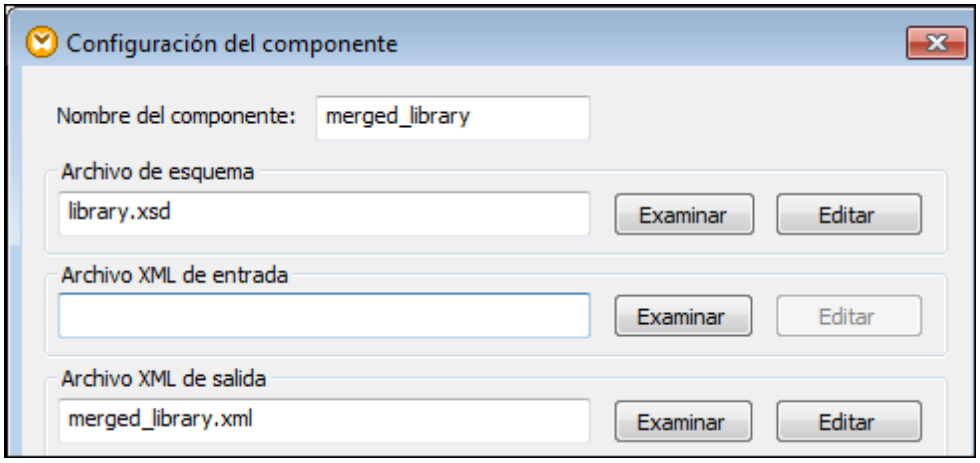
En el paso anterior tomamos el componente de destino y lo pegamos como componente de origen. Esto significa que el nuevo componente de origen hereda la configuración del componente de destino. Ahora debemos modificar el nombre de los archivos de instancia de entrada y salida para que sean correctos. Haga doble clic en el título de cada componente y, en el cuadro de diálogo "Configuración del componente", verifique y corrija el nombre de los archivos de entrada y salida de cada uno de ellos.



*Configuración del primer componente de origen (**books**)*



*Configuración del segundo componente de origen (**library**)*

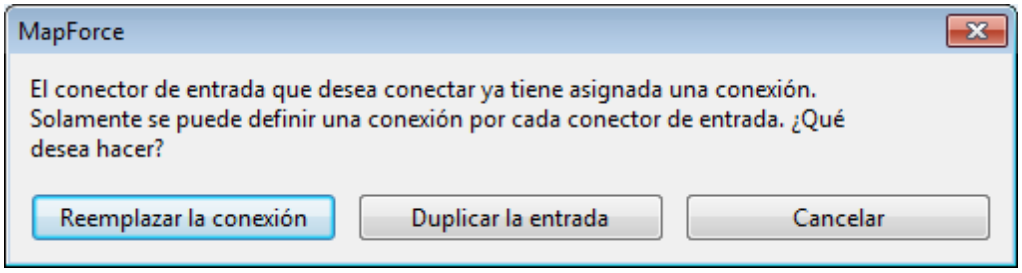


*Configuración del componente de destino (**merged_library**)*

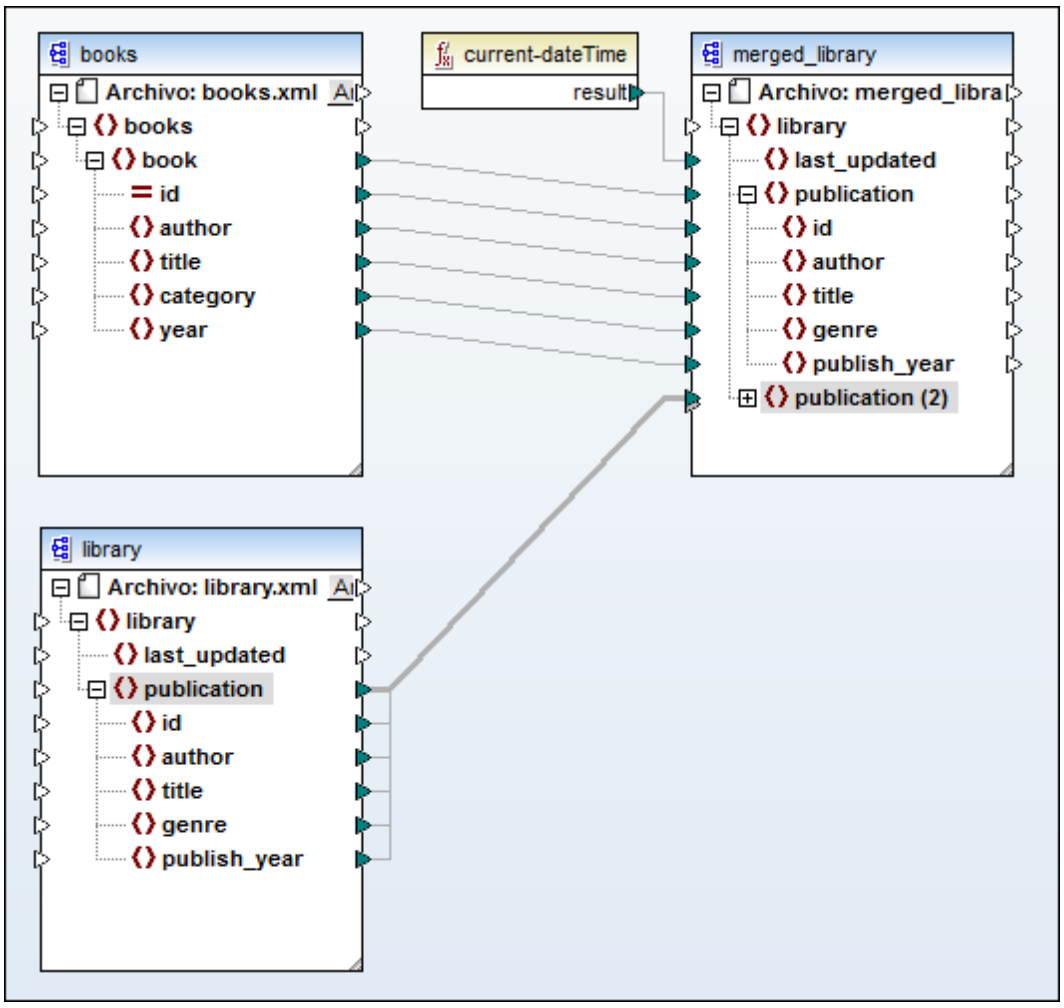
Como puede ver en las imágenes el primer componente de origen lee datos del archivo **books.xml**, mientras que el segundo lee datos de **library.xml**. Por su parte, el componente de destino genera datos en un archivo llamado **merged_library.xml**.

Paso nº4: crear las conexiones

El siguiente paso es dar la orden de escribir datos del segundo componente de origen en el componente de destino. Haga clic en el conector de salida del elemento `publication` del componente de origen **library** y arrástrelo hasta el conector de entrada del elemento `publication` del componente de destino **library**. Como el conector de entrada del destino ya está conectado, aparece esta notificación:



En este ejemplo concreto no queremos reemplazar la conexión porque nuestro objetivo es asignar datos procedentes de dos fuentes distintas. Por tanto, en este ejemplo elegimos la opción **Duplicar la entrada**. El resultado es que el componente de destino también aceptará datos del nuevo componente de origen. Este es el aspecto que tiene ahora el diseño de asignación:



Observe que el elemento `publication` del componente de destino está duplicado. El nodo `publication(2)` nuevo aceptará datos del componente de origen **library**. Y, lo que es más importante, aunque el nombre de este nodo aparezca como `publication(2)` en la asignación,

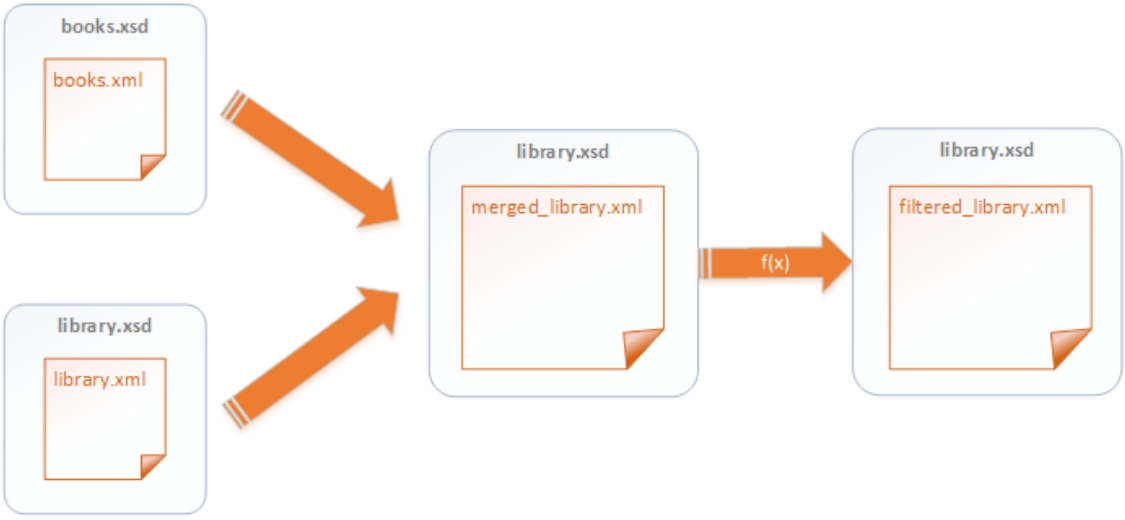
en el archivo XML resultante se llamará `publication`.

Ahora puede hacer clic en el botón **Resultados** situado en la parte inferior del panel de asignación para ver el resultado. Observe que tanto los datos de **library.xml** como los de **books.xml** se incluyeron en el archivo de salida **merged_library.xml**.

3.3 Trabajar con varios esquemas de destino

En el tutorial anterior aprendimos a asignar datos de varios esquemas de origen a un solo esquema de destino y creamos un archivo llamado **merged_library.xml** donde se almacenan los registros de libros procedentes de dos fuentes. Ahora imaginemos que alguien de otro departamento solicita un subconjunto de datos de este archivo XML. Concretamente solicita un archivo XML que contenga solamente los libros publicados después de 1900.

Si quiere puede modificar la asignación **MultipleSourcesToOneTarget.mfd** que viene con MapForce para poder generar tanto la biblioteca XML completa como la biblioteca con los datos filtrados.



Modelo abstracto de la transformación de datos

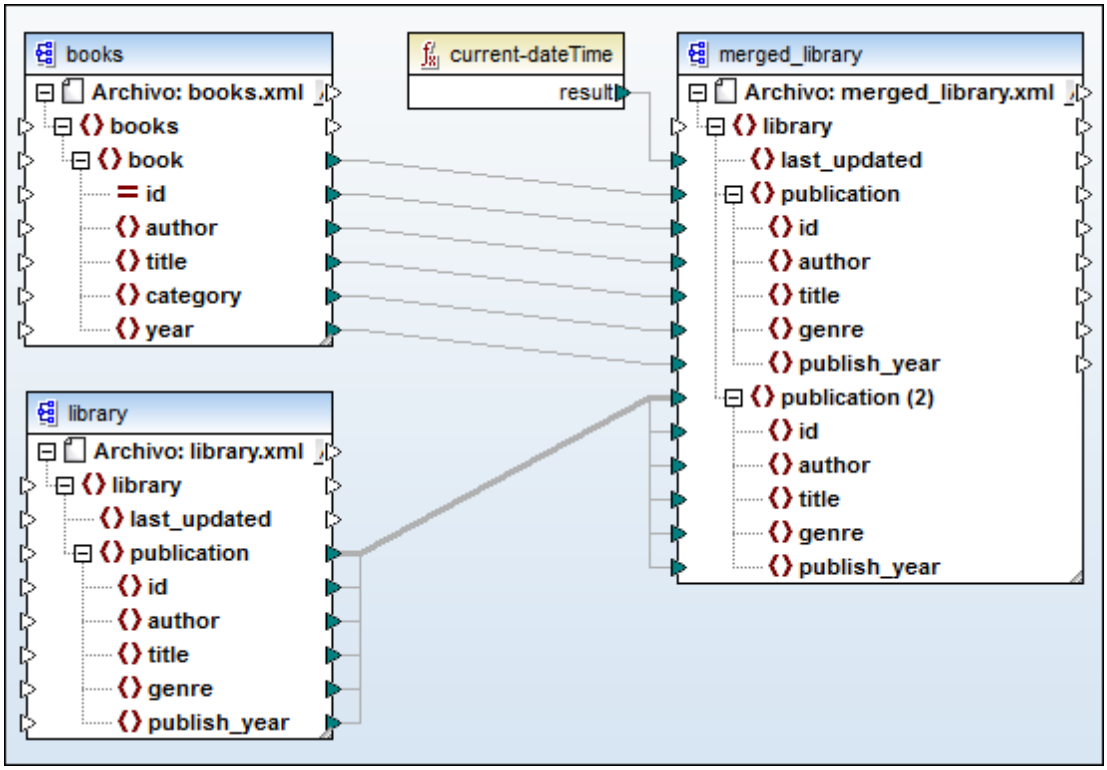
En este diagrama puede ver que primero se combinan lo datos de dos esquemas diferentes (**books.xsd** y **library.xsd**) y se guardan en un solo archivo XML llamado **merged_library.xml**. Después se transforman los datos con ayuda de una función de filtrado y se envían al siguiente componente, que se encarga de generar un archivo XML llamado **filtered_library.xml**. El componente intermedio hace tanto de destino como de origen. En MapForce esta técnica recibe el nombre de *encadenar asignaciones* y también se utiliza en este tutorial.

En este tutorial nuestro objetivo es poder generar tanto el archivo **merged_library.xml** como el archivo **filtered_library.xml** siempre que se quiera. Para conseguir nuestro objetivo es necesario seguir estos pasos:

Paso nº1: preparar el archivo de diseño de asignación

Como partida utilizamos la asignación **MultipleSourcesToOneTarget.mfd** de la carpeta `<Documentos>\Altova\MapForce2018\MapForceExamples\Tutorial\`. Esta asignación es la que diseñamos en el tutorial *Asignar varios orígenes de datos a un destino*. Para empezar, abra el archivo **MultipleSourcesToOneTarget.mfd** en MapForce y guárdelo con otro nombre.

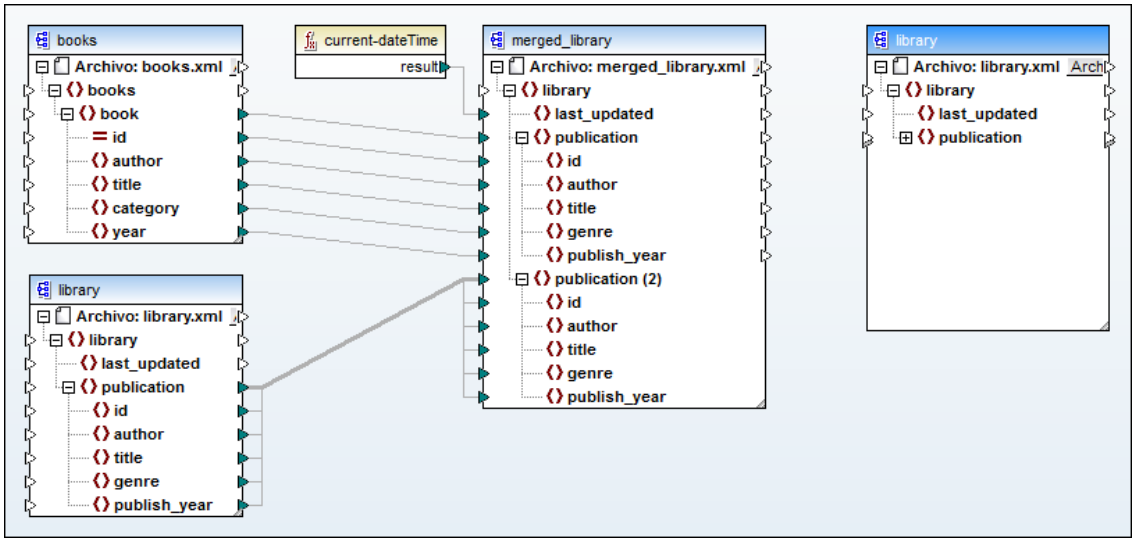
Asegúrese de guardar la asignación nueva en la carpeta `<Documentos>\Altova\MapForce2018\MapForceExamples\Tutorial\` porque hace referencia a varios archivos que están en esa carpeta.



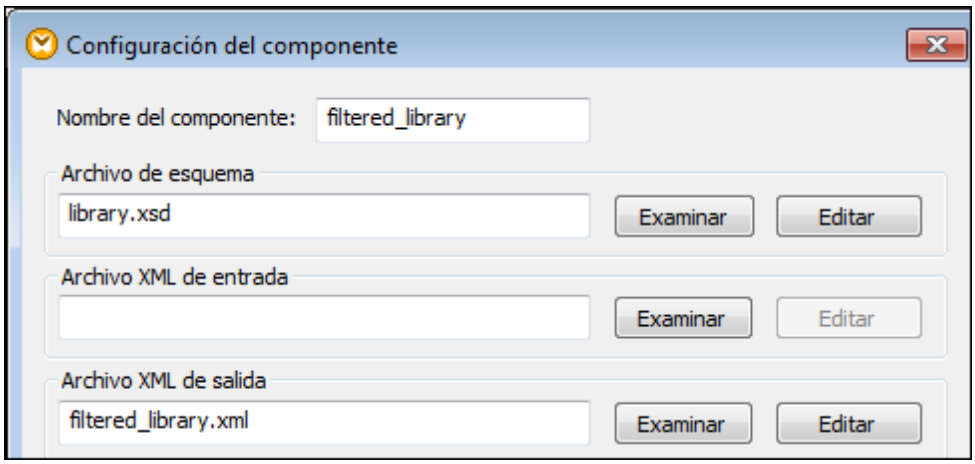
MultipleSourcesToOneTarget.mfd

Paso nº2: agregar y configurar el segundo componente de destino

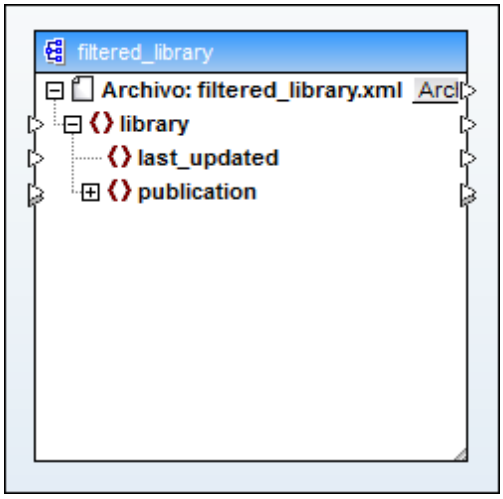
Para agregar otro componente de destino haga clic en el botón **Insertar archivo o esquema XML** de la barra de herramientas y abra el archivo **library.xsd** situado en la carpeta `<Documentos>\Altova\MapForce2018\MapForceExamples\Tutorial\`. Haga clic en Omitir cuando la aplicación solicite un archivo de instancia de muestra. La asignación tendrá este aspecto:



Como se aprecia en la imagen, ahora la asignación tiene dos componentes de origen (**books** y **library**) y dos componentes de destino. Para distinguirlos bien pondremos el nombre **filtered_library** al segundo componente de destino y definiremos el nombre del archivo XML que debe generar. Esto se consigue haciendo doble clic en el título del segundo componente de destino y editando su configuración:

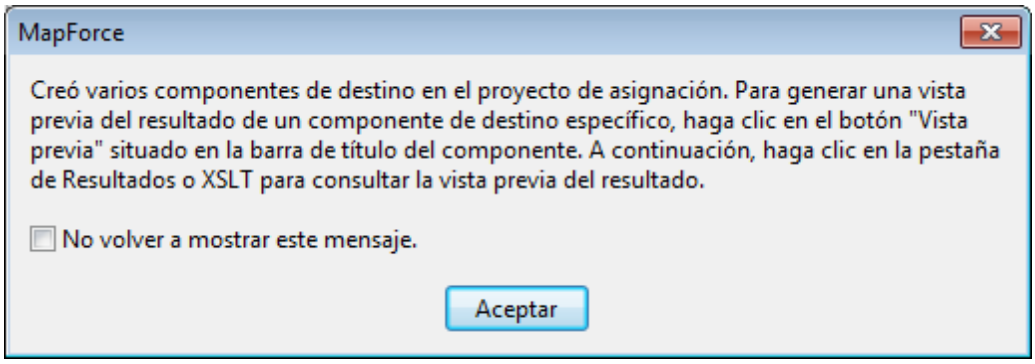


Observe que ahora el componente nuevo se llama **filtered_library** y que su archivo XML de salida se llama **filtered_library.xml**.

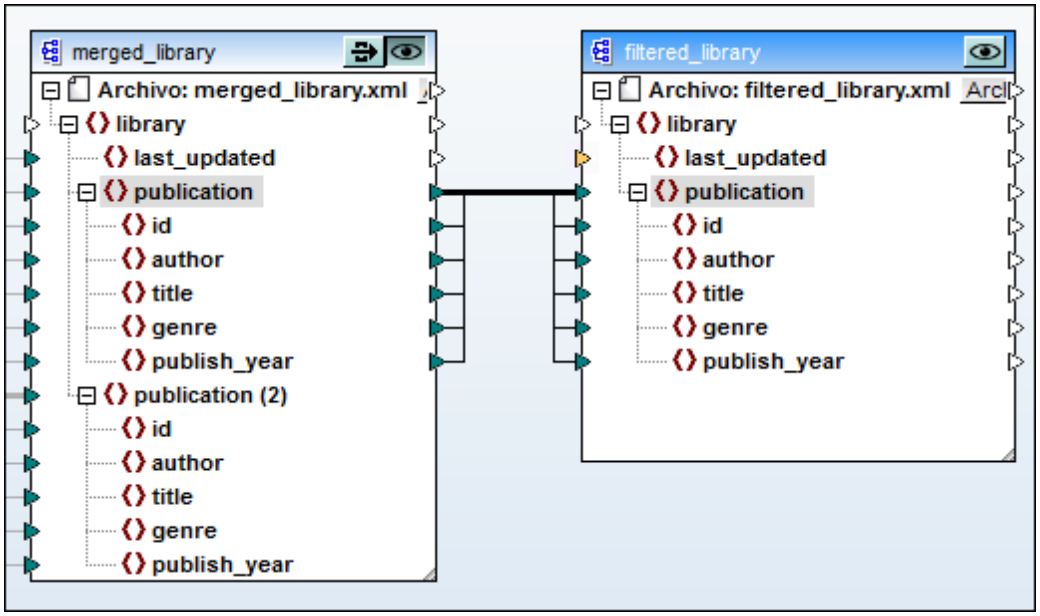


Paso nº3: crear las conexiones

Cree una conexión entre el elemento **publication** del componente **merged_library** y el elemento **publication** del componente **filtered_library**. Aparece una notificación:

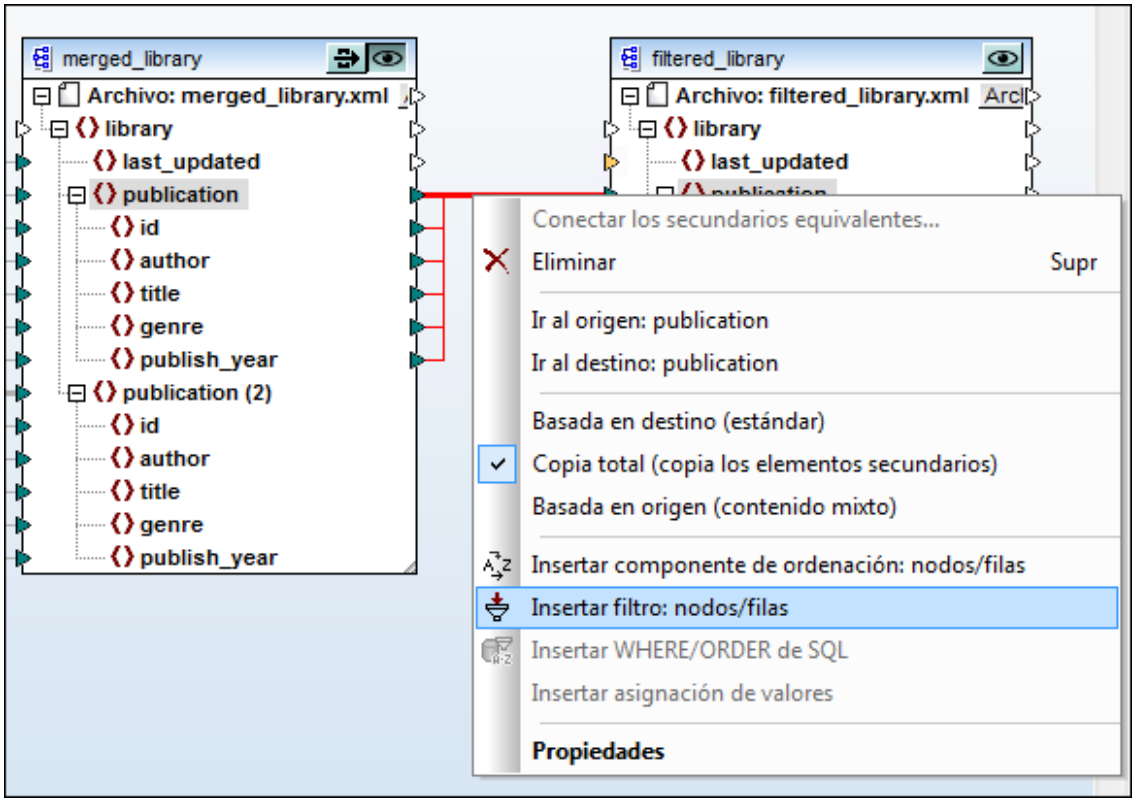


Haga clic en **Aceptar**. Observe que en los componentes de destino ahora aparecen nuevos botones: **Vista previa** ( ) y **Paso a través** ( ). En los siguientes pasos del tutorial aprenderemos a usar estos botones.

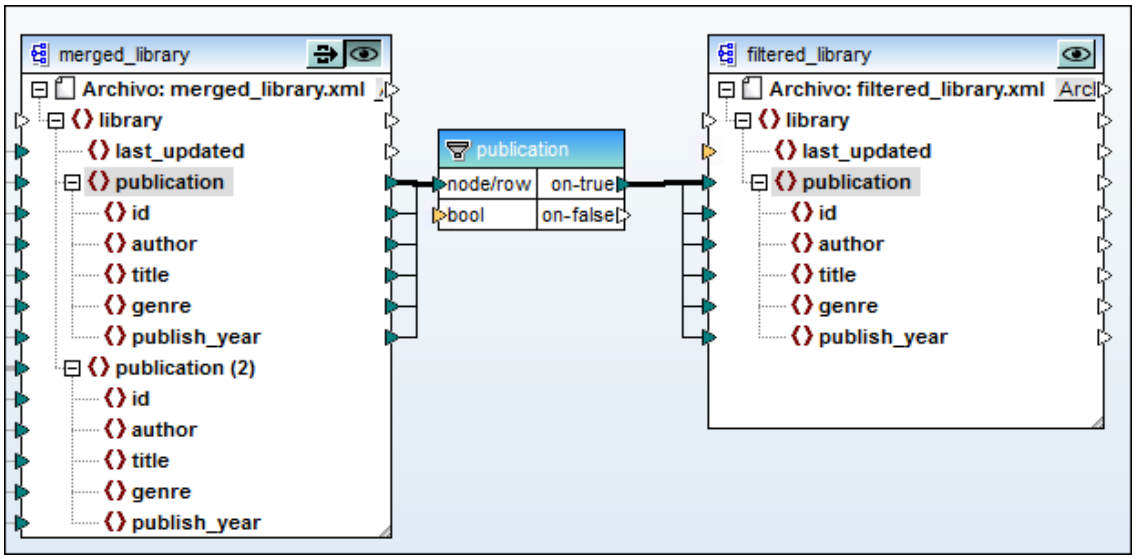


Paso nº4: filtrar los datos

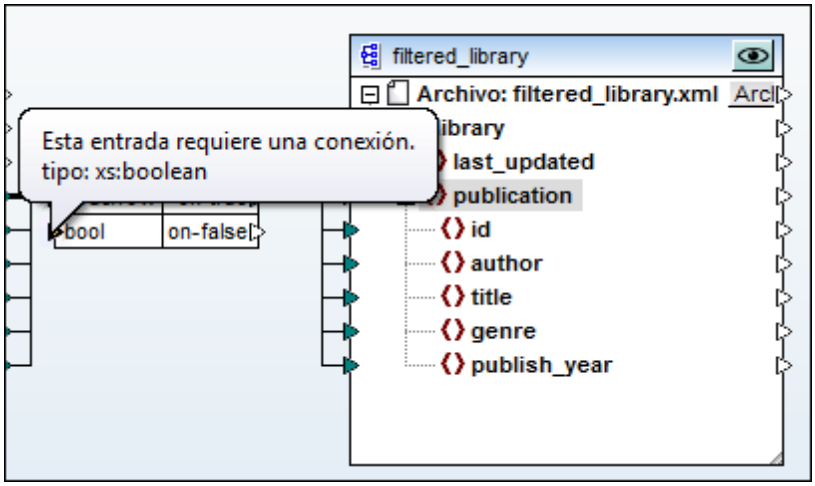
Para filtrar los datos antes de pasarlos a **filtered_library** usaremos un componente de filtrado. Para agregar un componente de filtrado haga clic con el botón derecho en la conexión que existe entre **merged_library** y **filtered_library** y seleccione el comando **Insertar filtro: nodos/filas** del menú contextual.



El componente de filtrado aparece ahora en la asignación.



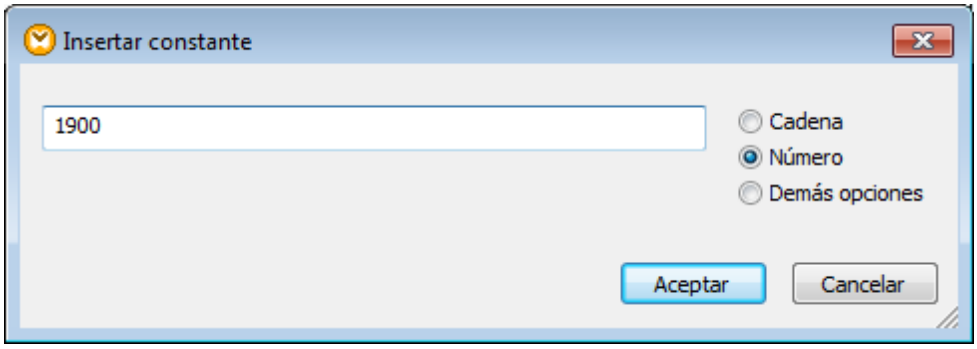
Como se aprecia en la imagen, el conector de entrada bool es de color naranja, lo cual indica que exige una entrada. Si pasa el puntero por el conector podrá ver que exige una entrada de tipo `xs:boolean`. Recuerde que para poder ver la información rápida es necesario tener habilitado el botón **Mostrar información rápida** de la barra de herramientas.



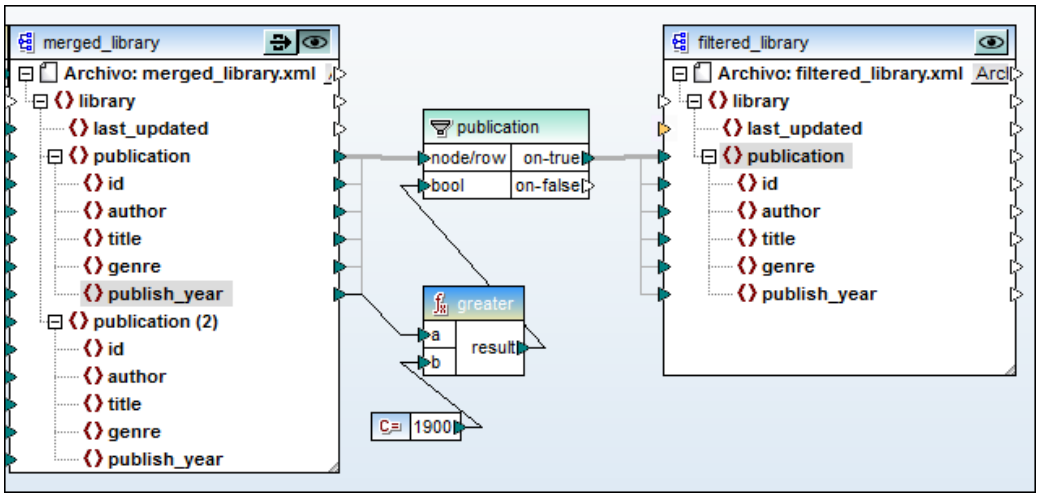
El componente de filtrado exige una condición que devuelva el valor `true` o `false`. Cuando la condición booleana devuelva `true`, los datos de la secuencia **publication** actual se copiarán en el componente de destino. Sin embargo, cuando devuelva `false`, no se copiarán datos.

En este tutorial nuestro objetivo es filtrar todos los libros que se publicaron después de 1900. Para crear la condición:

1. Añada una constante de tipo numérico que tenga el valor "1900" (haciendo clic en **Insertar | Constante**). Elija el tipo *Número*.



2. En la ventana Bibliotecas busque la función **`greater`** y arrástrela hasta el área de asignación.
3. Cree conexiones con la función **`greater`** tal y como muestra la siguiente imagen. Estas conexiones dan a MapForce esta instrucción: "cuando `publish_year` sea mayor que 1900, copiar el elemento de origen `publication` actual en el elemento de destino `publication`".



Paso nº5: consultar la vista previa de resultados y guardar los resultados de cada componente de destino

Ahora ya puede consultar la vista previa de resultados y guardar los resultados de ambos componentes de destino. Cuando en la misma asignación hay varios componentes de destino, puede elegir cuál de los dos se usa para generar la vista previa de resultados. Esto se hace con el botón **Vista previa** de los componentes. Si el botón **Vista previa** de un componente está pulsado, significa que ese componente concreto se usará para generar la vista previa de resultados. Este botón no puede estar pulsado en más de un componente a la vez.

Por tanto, cuando quiera consultar y guardar los resultados de **merged_library** (es decir, del componente intermedio), siga estos pasos:

1. Haga clic en el botón **Vista previa** del componente **merged_library**.
2. Haga clic en el botón **Resultados** situado en la parte inferior del panel de asignación.
3. En el menú **Resultados** haga clic en el comando **Guardar el archivo de salida** si quiere guardar los resultados en un archivo.

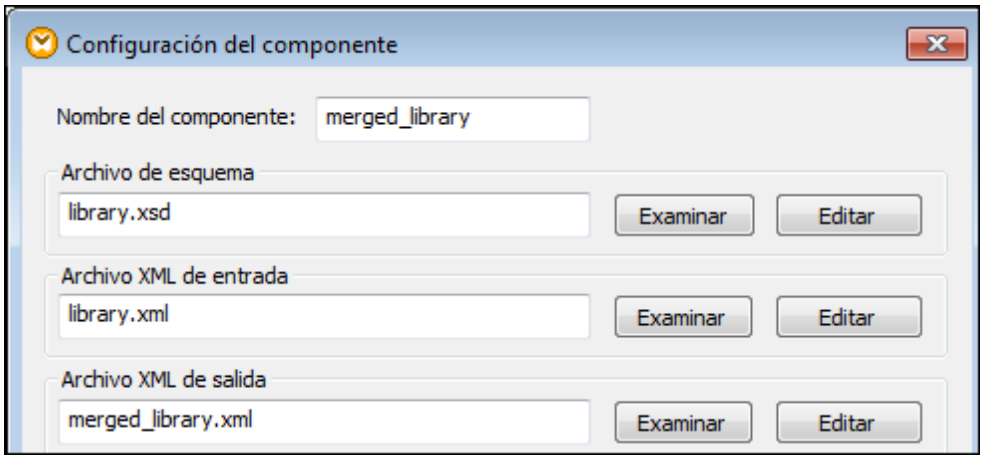
Cuando quiera cuando quiera consultar y guardar los resultados del componente **filtered_library**:

1. Haga clic en el botón **Paso a través** del componente **merged_lilbrary**.
2. Haga clic en el botón **Vista previa** del componente **filtered_library**.
3. Haga clic en el botón **Resultados** situado en la parte inferior del panel de asignación.
4. En el menú **Resultados** haga clic en el comando **Guardar el archivo de salida** si quiere guardar los resultados en un archivo.

El estado del botón **Paso a través** desempeña un papel importante en las asignaciones que tienen varios componentes de destino. Si este botón está pulsado, MapForce deja pasar los datos por el componente intermedio para que pueda consultar la vista previa de los resultados de toda la asignación.

Si el botón **Paso a través** no está pulsado, la vista previa incluirá solamente los resultados de la asignación de datos entre **merged_library** y **filtered_library**. En el ejemplo que nos

ocupa esto daría lugar a un error porque el componente intermedio no tiene un archivo XML de entrada válido para leer datos. Para resolver este problema haga doble clic en el título del componente y especifique un archivo XML de entrada válido:

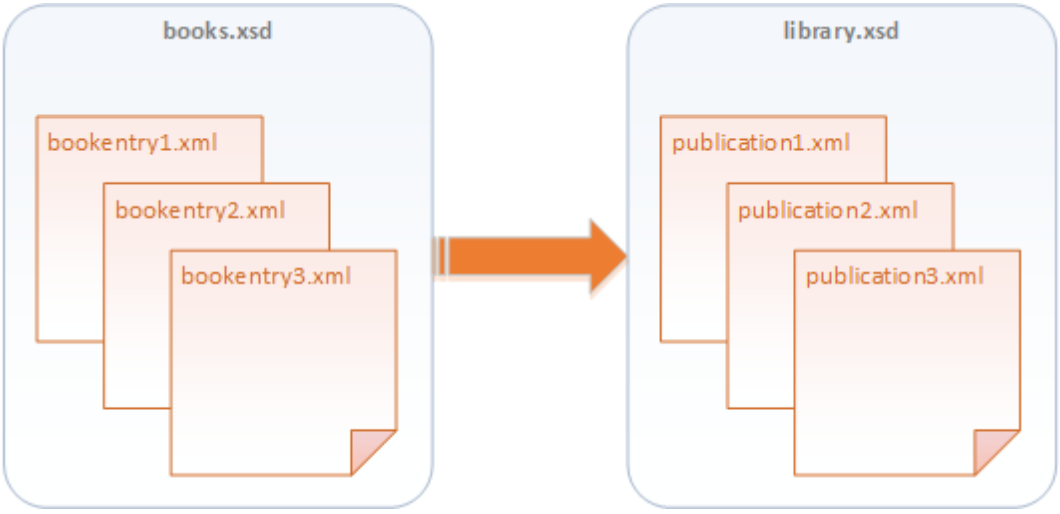


La asignación creada en este tutorial tiene varios componentes de salida y podemos consultar y guardar los resultados de cada uno de ellos. Para más información sobre el funcionamiento de los componentes de paso a través consulte el apartado *Asignaciones encadenadas*.

3.4 Procesar y generar archivos de forma dinámica

Este tutorial explica cómo leer datos de varios archivos XML de origen y escribirlos en varios archivos de destino en la misma transformación. Para comprender esta característica crearemos una asignación cuyos objetivos serán:

1. Leer datos de varios archivos XML ubicados en el mismo directorio.
2. Pasar cada archivo XML a un esquema XML nuevo.
3. Por cada archivo XML de origen generar un archivo XML de destino nuevo bajo un esquema nuevo.
4. Eliminar la declaración XML y la declaración de espacio de nombres de los archivos generados.



Modelo abstracto de la transformación de datos

Como ejemplo usaremos tres archivos XML de origen, ubicados en la carpeta `<Documentos>\Altova\MapForce2018\MapForceExamples\Tutorial\`, que se llaman **bookentry1.xml**, **bookentry2.xml** y **bookentry3.xml**. Cada uno de ellos contiene los datos de un solo libro.

```
<?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>
```

bookentry1.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>
```

bookentry2.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>
```

bookentry3.xml

Los tres archivos XML de origen usan el esquema **books.xsd**, ubicado en la carpeta `<Documentos>\Altova\MapForce2018\MapForceExamples\Tutorial\`. Para pasar los archivos de origen a otro esquema XML usaremos el esquema **library.xsd** (ubicado en la misma carpeta). Una vez completada la transformación, la asignación generará tres archivos basados en este nuevo esquema (consulte los fragmentos de código que aparecen a continuación). También configuraremos la asignación para que los nombres de los archivos de salida sean **publication1.xml**, **publication2.xml** y **publication3.xml**. Recuerde que debemos eliminar la declaración XML y la declaración de espacio de nombres.

```
<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>
```

publication1.xml

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

publication2.xml

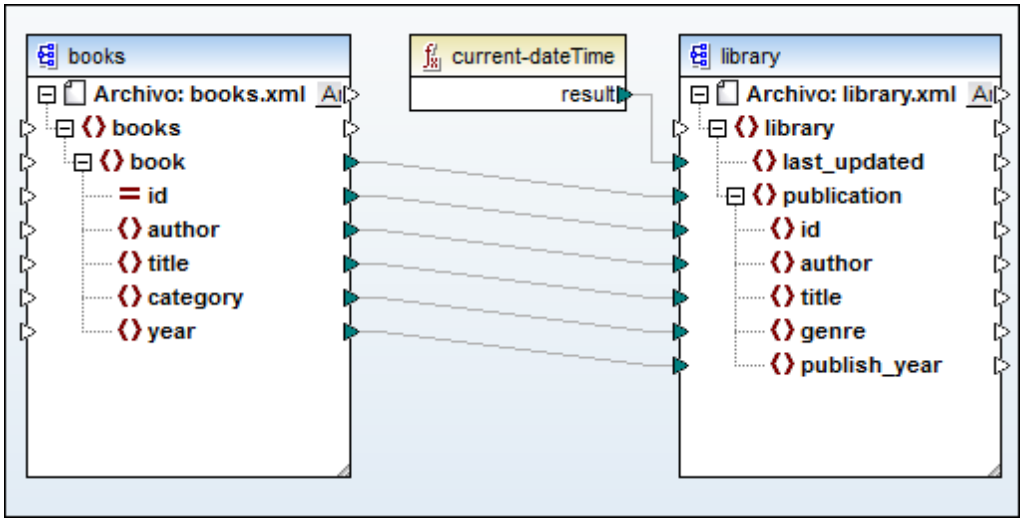
```
<library>
   <publication>
      <id>3</id>
      <author>Herman Melville</author>
      <title>Moby Dick</title>
      <genre>Fiction</genre>
      <publish_year>1851</publish_year>
   </publication>
</library>
```

publication3.xml

Paso nº1: preparar el archivo del diseño de asignación

Como punto de partida usamos la asignación **BooksToLibrary.mfd** de la carpeta `<Documentos>\Altova\MapForce2018\MapForceExamples\Tutorial\`. Se trata de la asignación que diseñamos en el tutorial *Pasar datos XML a un esquema nuevo*. Para empezar abra el archivo **BooksToLibrary.mfd** en MapForce y guárdelo en la misma carpeta con otro nombre.

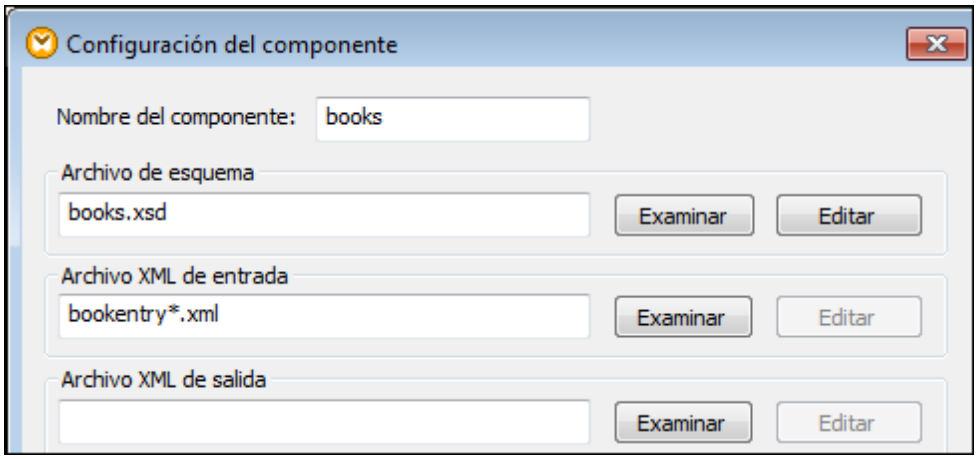
> Asegúrese de guardar la asignación nueva en la carpeta `<Documentos>\Altova\MapForce2018\MapForceExamples\Tutorial\` porque hace referencia a varios archivos que están en esa carpeta.



BooksToLibrary.mfd

Paso nº2: configurar la entrada

Para que MapForce procese varios archivos de instancia XML haga doble clic en el título del componente de origen y escriba **bookentry*.xml** en el campo *Archivo XML de entrada* del cuadro de diálogo "Configuración del componente".



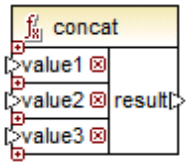
Cuadro de diálogo "Configuración del componente"

El carácter comodín asterisco (*) del nombre de archivo indica que MapForce debe usar todos los archivos que tengan el prefijo **bookentry-** como entrada para la asignación. Como la ruta de acceso es relativa, MapForce buscará todos los archivos **bookentry-** que estén en el directorio del archivo de asignación. De todas maneras, también puede usar el carácter comodín * con rutas de acceso absolutas.

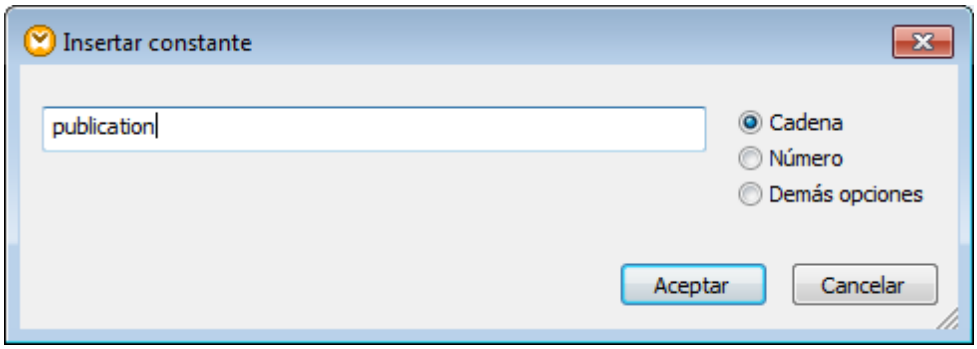
Paso nº3: configurar la salida

Para crear el nombre de archivo de cada archivo de salida usaremos la función **concat**. Esta función concatena (une) todos los valores que recibe como argumento.

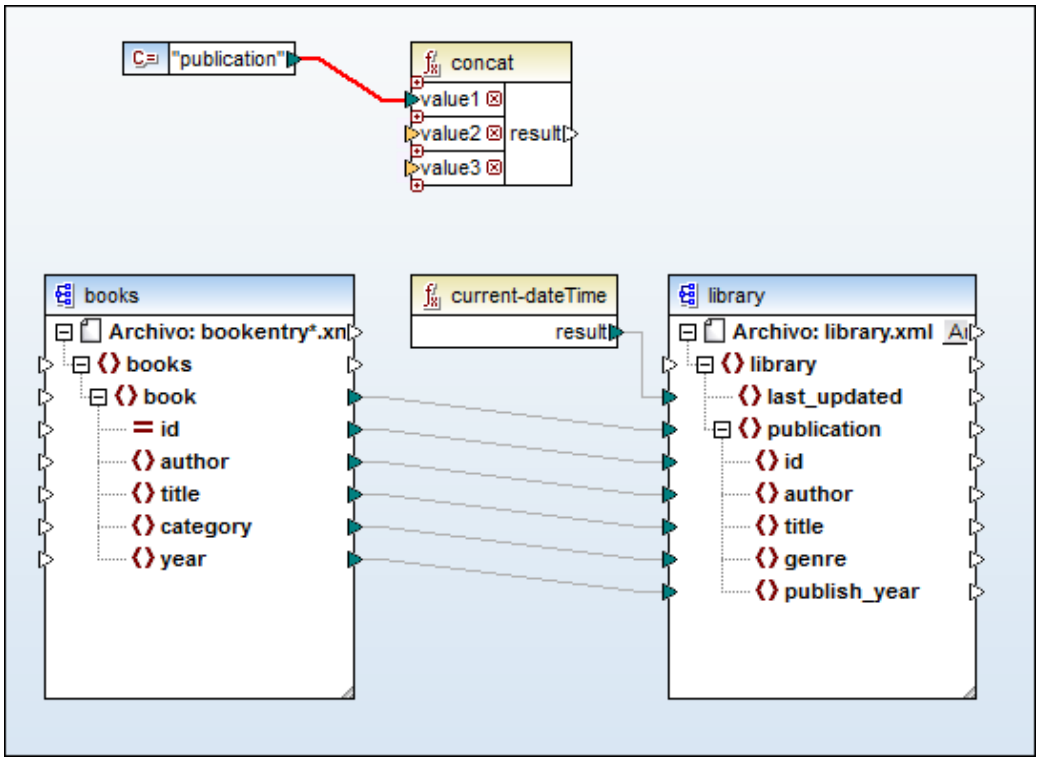
1. Busque la función **concat** en la ventana Bibliotecas y arrástrela hasta el área de asignación. Esta función se añade a la asignación con dos parámetros predeterminados pero puede añadirle más parámetros si lo necesita. Haga clic en el símbolo **Agregar parámetro** (⊡) situado dentro de la función y añádale otro parámetro más. El símbolo **Eliminar parámetro** (⊠) sirve para eliminar parámetros.



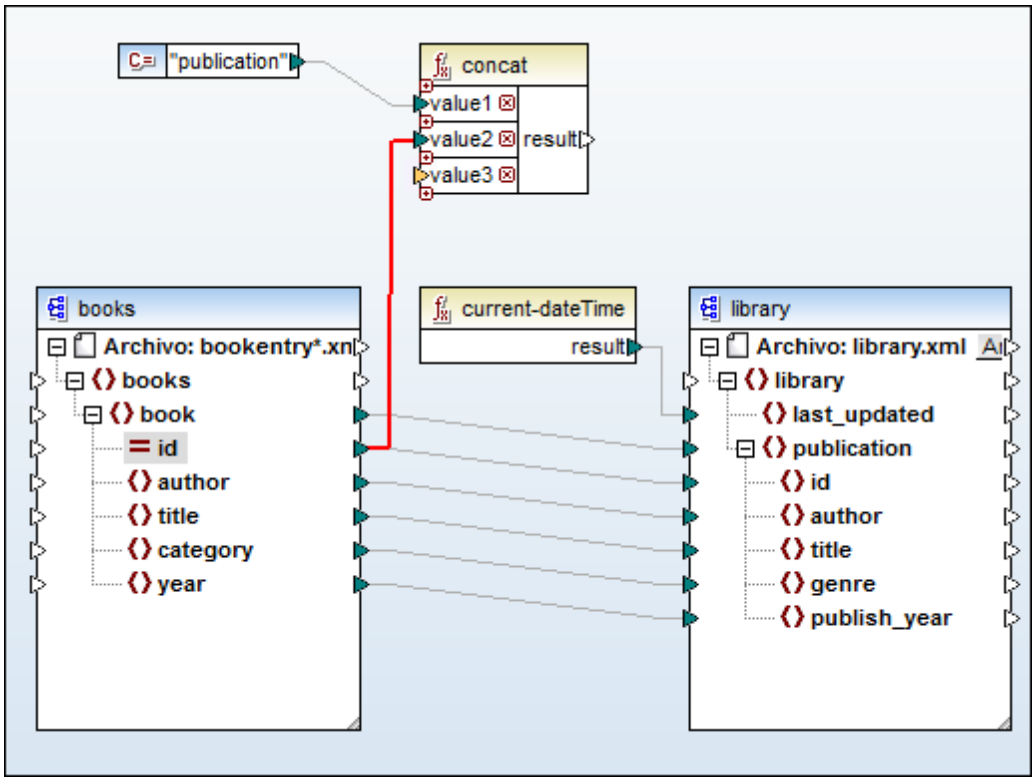
2. Inserte una constante (con el comando de menú **Insertar | Constante**) que tenga el valor **publication** y con el tipo *Cadena*.



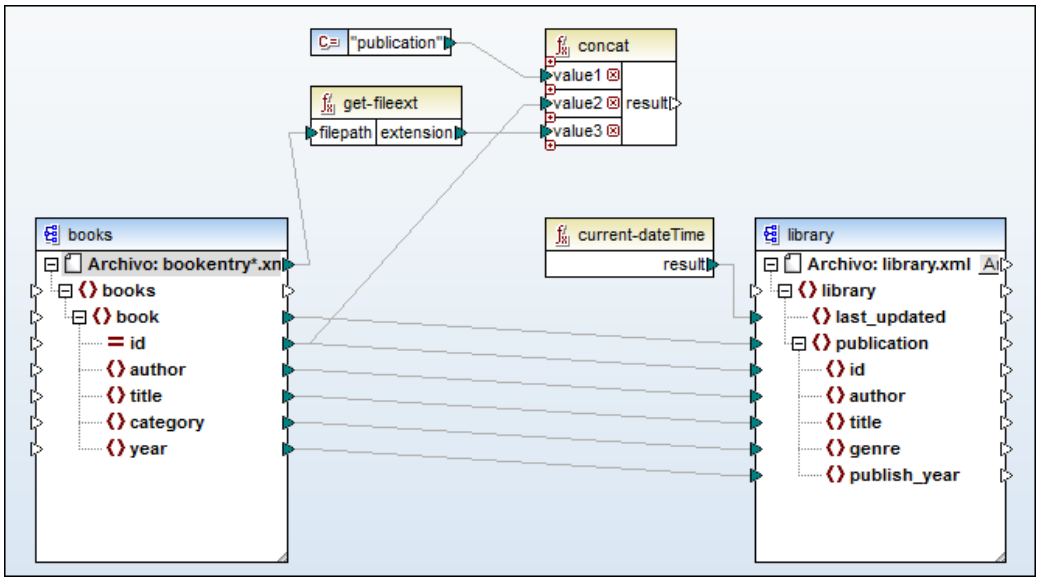
3. Conecte la constante con el valor **value1** de la función **concat**.



4. Conecte el atributo `id` del componente de origen con el valor **value2** de la función **`concat`**.



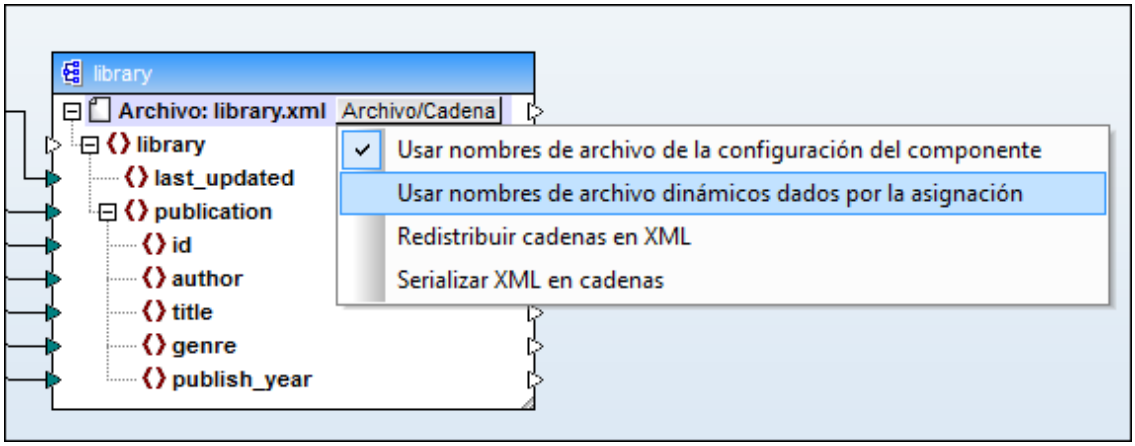
5. Ahora busque la función **get-fileext** en la ventana Bibliotecas y arrástrela hasta el área de asignación. Cree una conexión entre el nodo de nivel superior del componente de origen (**Archivo: books.xml**) y el parámetro **filepath** de esta función. Después cree una conexión entre el resultado **result** de la función **get-fileext** y el valor **value3** de la función **concat**. De este modo se extrae solamente la extensión (en este caso `.xml`) del nombre del archivo de origen.



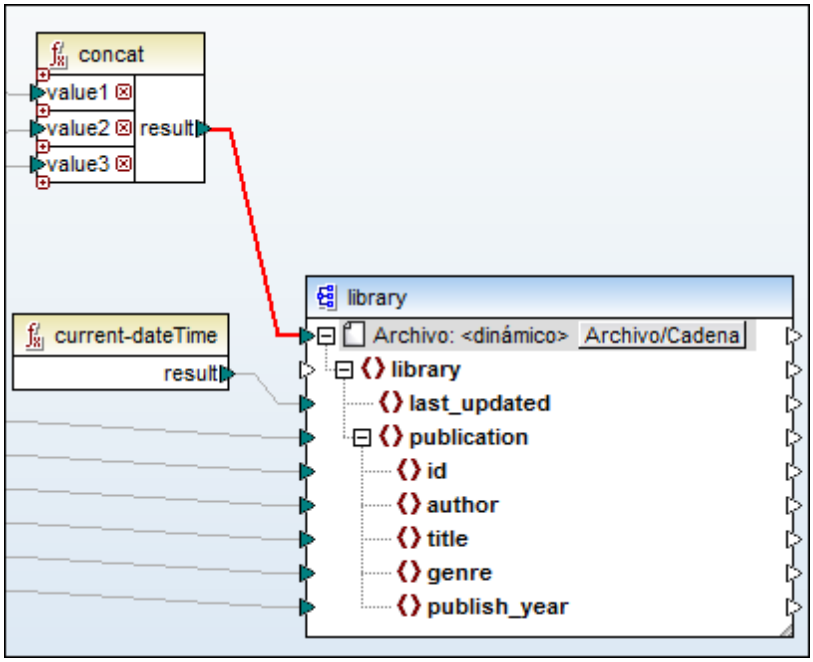
Llegados a este punto la función **concat** recibe como parámetros tres valores, que tras concatenarlos, crearán en nombre de los archivos de salida (p. ej. **publication1.xml**):

Partes del nombre de archivo de salida	Ejemplo
La constante **publication** aporta el valor de cadena constante `publication`.	publication
El atributo `id` del archivo XML de origen aporta un valor de identificador único para cada archivo. Esto evita que todos los archivos de salida tengan el mismo nombre.	1
La función **get-fileext** devuelve la extensión del nombre de archivo que se debe generar.	.xml

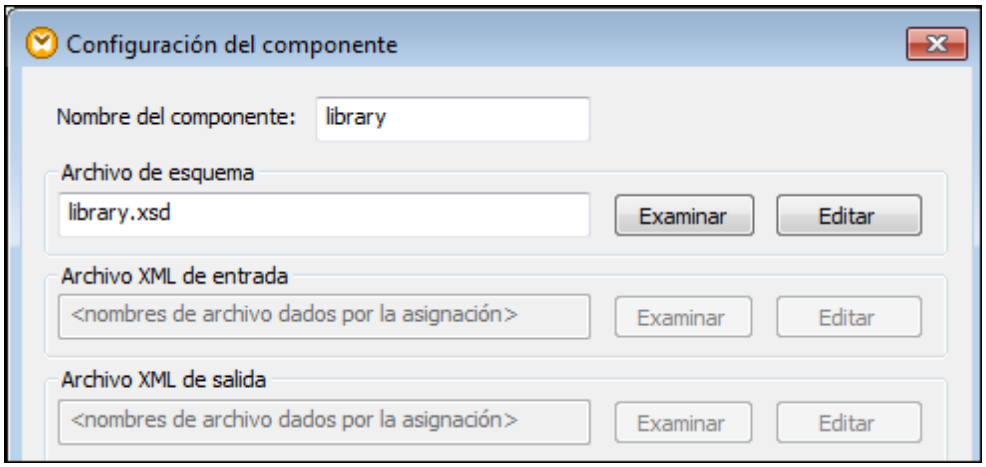
Ahora debemos dar a MapForce la orden de generar el nombre de archivo cuando se ejecute la asignación. Para ello haga clic en el botón Archivo o Archivo/Cadena del componente de destino y seleccione el comando **Usar nombres de archivo dinámicos dados por la asignación**.



MapForce ya tiene la orden de generar los archivos de instancia de forma dinámica, con el nombre dado por la asignación. En nuestro ejemplo el nombre se crea con la función **concat**. Por tanto, conectaremos el resultado de la función **concat** con el nodo **Archivo: <dinámico>** del componente de destino.

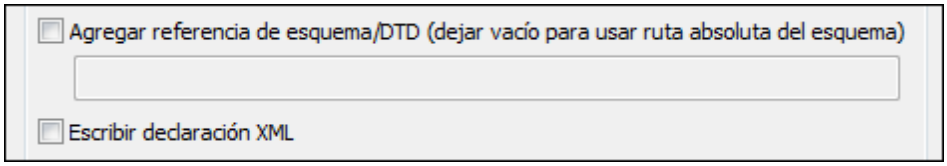


Si hace doble clic en el título del componente de destino, observará que los campos *Archivo XML de entrada* y *Archivo XML de salida* están deshabilitados y que en el cuadro de texto aparece *<nombres de archivos dados por la asignación>*.

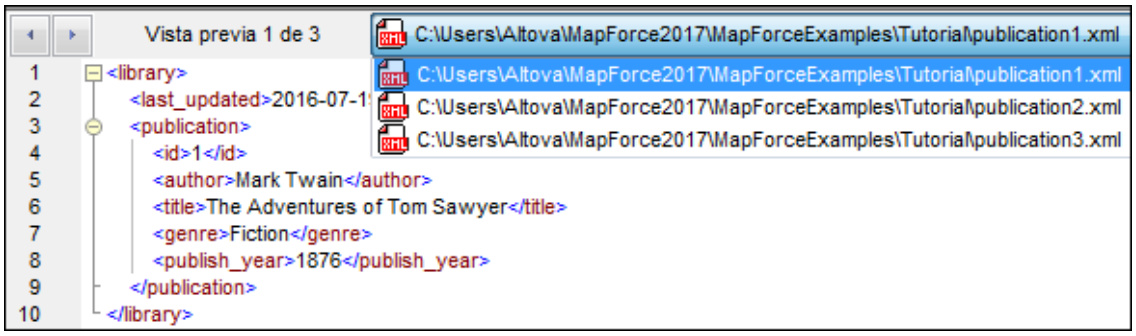


Esto nos indica que los nombres de los archivos de instancia se obtienen de forma dinámica de la asignación y por ello ya no es necesario definirlos en la configuración del componente.

Por último, eliminaremos la declaración de esquema y de espacio de nombres del componente de destino. Para ello desactivamos las casillas *Agregar referencia de esquema/DTD...* y *Escribir declaración XML* en el cuadro de diálogo "Configuración del componente".



Ahora puede ejecutar la asignación y consultar una vista previa de lo resultados y el nombre de los archivos que se generarán. Esta asignación genera varios archivos de salida y puede navegar por ellos con los botones de flecha situados en la esquina superior izquierda del panel *Resultados* (o seleccionando un archivo en la lista desplegable).



Altova MapForce 2018 Basic Edition

Tareas generales

4 Tareas generales

Esta sección del tutorial describe tareas y conceptos generales de MapForce, como trabajar con asignaciones, componentes y conexiones.

4.1 Trabajar con asignaciones 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 a otro. Una asignación está formada por componentes, que se van añadiendo al área de asignación de MapForce para crear las transformaciones de datos (p. ej. para convertir documentos XML que siguen un esquema en documentos que siguen otro esquema). 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. Los diseños de asignación de MapForce tienen la extensión de archivo **.mfd**.

Para crear una asignación nueva:

1. Tiene dos opciones:
   - o hacer clic en el comando de menú **Archivo | Nuevo** o
   - o hacer clic en el botón **Nuevo** de la barra de herramientas.

La asignación se crea pero el área de asignación está vacía. Para ser válida la asignación necesita tener como mínimo dos componentes conectados, de modo que el siguiente paso consiste en agregar componentes a la asignación (*véase Agregar componentes a la asignación*) y dibujar conexiones entre ellos (*véase Trabajar con conexiones*).

4.1.1 Agregar componentes a la asignación

En MapForce un componente es el elemento gráfico que representa visualmente la estructura (esquema) de los datos o el elemento que determina cómo se deben transformar los datos (funciones). Los componentes son piezas fundamentales necesarias para construir una asignación. En el área de asignación de MapForce los componentes aparecen en forma de rectángulo. Estos son algunos ejemplos de componente:

- Constantes
- Filtros
- Condiciones
- Funciones
- Documentos EDI (UN/EDIFACT, ANSI X12, HL7)
- Archivos Excel 2007+
- Componentes de entrada simples
- Componentes de salida simples
- Esquemas XML y documentos DTD

Para agregar un componente a la asignación:

Hay varias maneras de agregar componentes a la asignación:

- En el menú **Insertar** haga clic en el comando que corresponda al tipo de componente

que desea agregar (p. ej. **Archivo o esquema XML**).
- Arrastre un archivo desde el explorador de Windows hasta el área de asignación. Tenga en cuenta que esta operación solo es válida para componentes basados en archivos compatibles.
- Haga clic en el botón correspondiente de la barra de herramientas **Insertar componente**.

*Barra de herramientas **Insertar componente** (MapForce Enterprise Edition)*

Cada tipo de componente tiene un objetivo y comportamiento concretos. En algunos casos MapForce ofrece un asistente que le guiará en el proceso de creación del componente. Por ejemplo, si añade un esquema XML, aparece un cuadro de diálogo donde tiene la opción de seleccionar también un archivo de instancia.

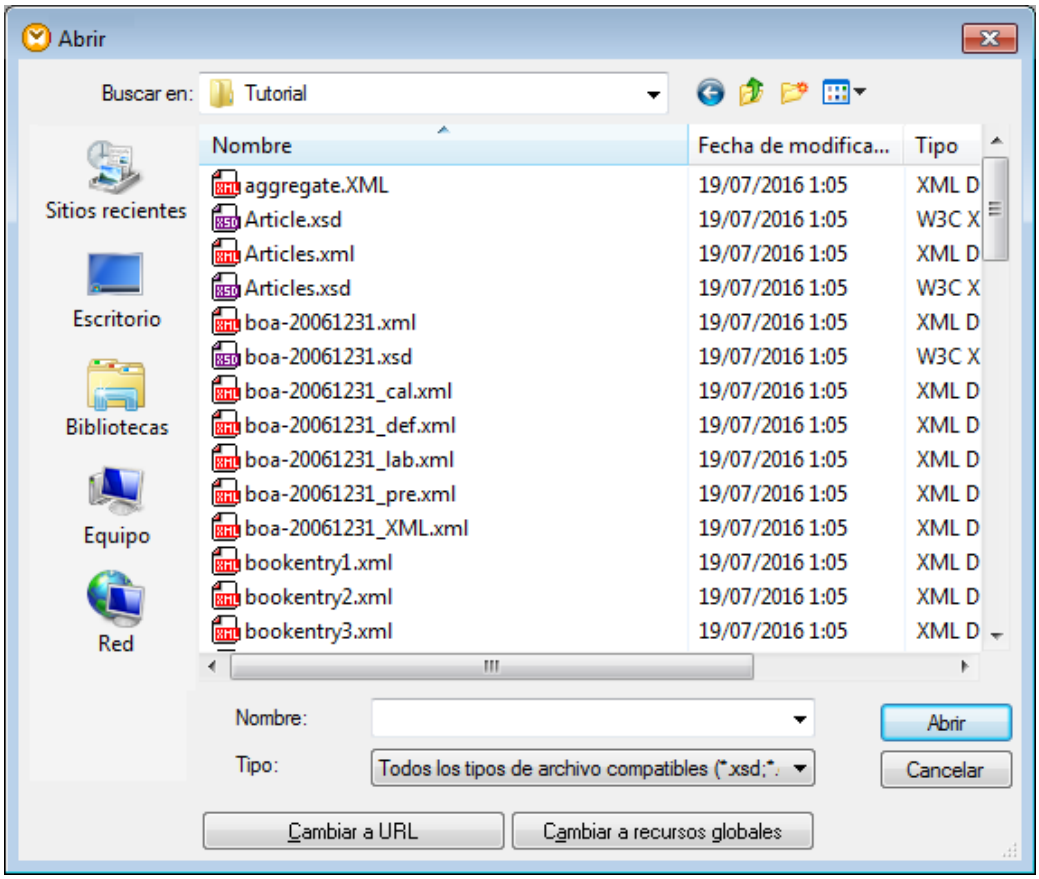
Para más información consulte el apartado Trabajar con componentes. Además, la sección Orígenes y destinos de datos explica en detalle todas las tecnologías compatibles que pueden utilizarse como origen o destino de las asignaciones. Y en la sección Diseño de asignaciones encontrará información sobre los componentes integrados de MapForce que sirven para almacenar datos temporalmente o transformarlos (p. ej. filtros y componentes de ordenación).

4.1.2 Agregar componentes desde una URL

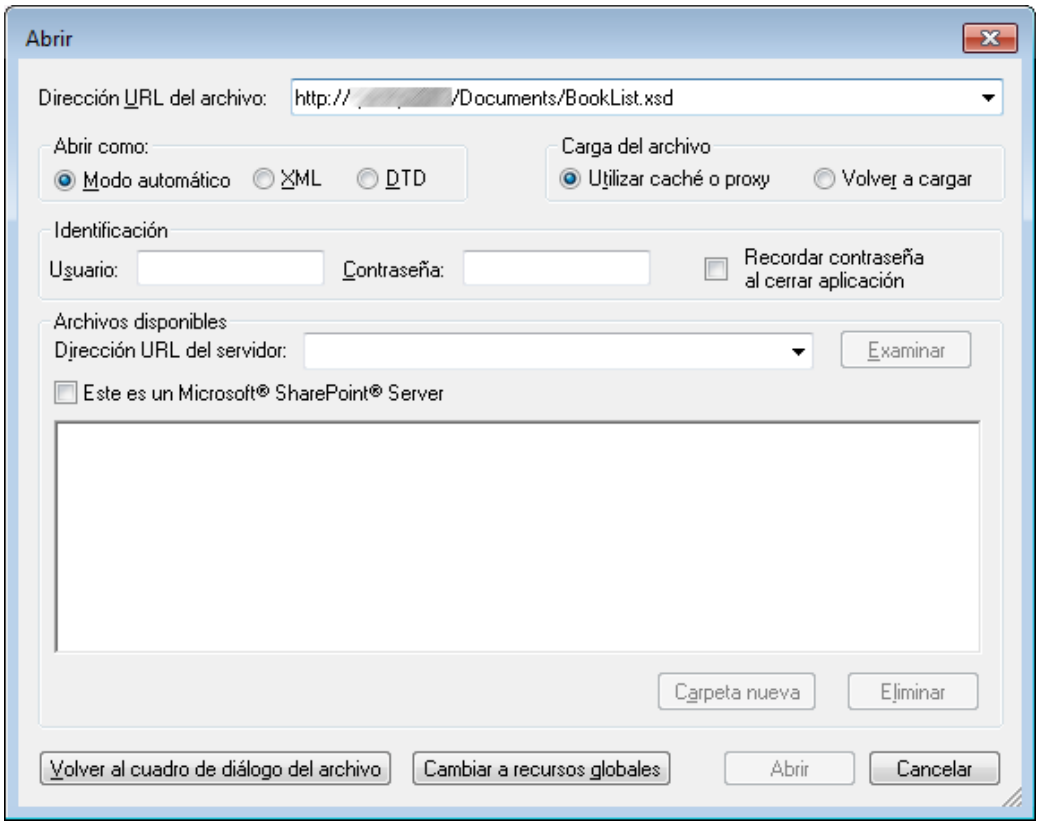
Los componentes de la asignación también pueden estar basados en archivos ubicados en una URL, no solo en archivos locales. Sin embargo, esta operación solamente es compatible con los componentes de origen (es decir, la asignación puede leer datos de archivos remotos). Los protocolos compatibles son HTTP, HTTPS y FTP.

Para agregar un componente desde una URL:

1. En el menú b seleccione el comando que corresponda al tipo de componente que desea agregar (p. ej. **Archivo o esquema XML**).
2. En el cuadro de diálogo "Abrir" haga clic en el botón **Cambiar a URL**.



3. Introduzca la URL del archivo en el cuadro de texto *Dirección URL del archivo:* y haga clic en **Abrir**.



Asegúrese que el tipo de archivo del campo *Dirección URL del archivo* coincide con el tipo de archivo especificado en el paso nº1.

Si el servidor requiere autenticación con contraseña, deberá introducir el nombre de usuario y la contraseña. Si desea que MapForce recuerde el nombre y la contraseña la próxima vez que se inicie, introduzca los datos de autenticación en el cuadro de diálogo "Abrir" y marque la casilla *Recordar contraseña al cerrar aplicación.*

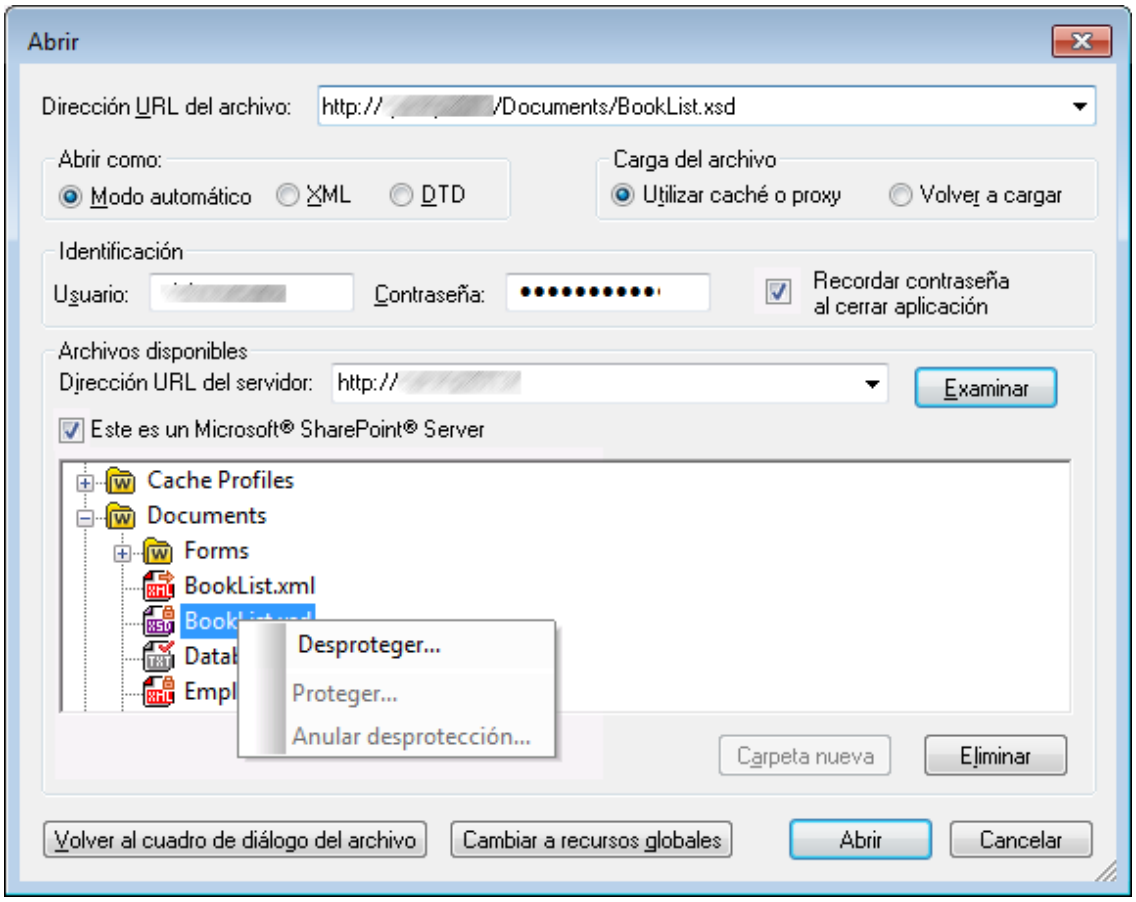
El grupo de opciones *Abrir como* sirven para definir la gramática utilizada por el analizador para abrir el archivo. La opción predeterminada y recomendada es *Modo automático*.

Si lo más probable es que el archivo que desea cargar no sufra modificaciones, seleccione la opción *Utilizar caché o proxy* para almacenar en caché los datos y así acelerar el proceso de carga del archivo. Sin embargo, si prefiere que el archivo se vuelva a cargar cada vez que abra la asignación, elija la opción *Volver a cargar*.

Si trabaja con un servidor WebDAV, introduzca la URL del servidor en el cuadro de texto *Dirección URL del servidor:* y haga en **Examinar** para explorar los archivos. Aunque la vista previa muestra todos los tipos de archivo, asegúrese de elegir el mismo tipo de archivo que especificó en el paso nº1 para evitar errores.

Si trabaja con un servidor Microsoft SharePoint Server, marque la casilla *Este es un Microsoft® SharePoint® Server*. Esto permite ver el estado de protección o desprotección de los archivos en

el panel de vista previa. Si quiere impedir que otros usuarios editen el archivo en el servidor mientras lo utiliza en MapForce, haga clic con el botón derecho en el archivo y seleccione el comando **Desproteger**. Para proteger un archivo desprotegido con anterioridad, haga clic con el botón derecho y seleccione **Proteger**.



Cuadro de diálogo "Abrir" (en el modo URL)

4.1.3 Seleccionar el lenguaje de transformación

Puede elegir entre dos lenguajes de transformación de datos:

- XSLT 1.0
- XSLT 2.0

Para elegir el lenguaje de transformación tiene dos opciones:

- En el menú **Resultados** haga clic en el nombre del lenguaje que desea usar para la transformación.
- En la barra de herramientas **Selección del lenguaje** 

haga clic en el nombre del lenguaje que desea usar.

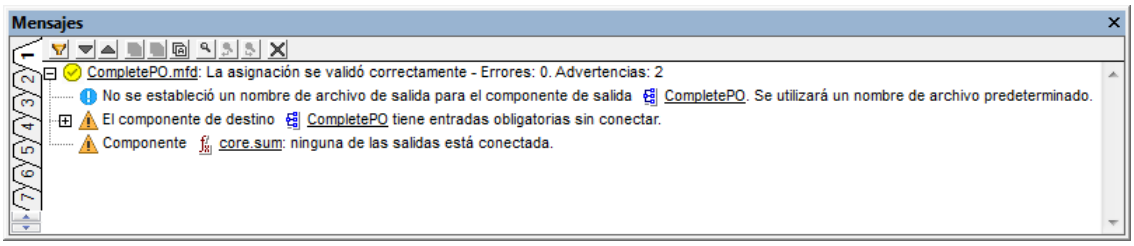
4.1.4 Validar la asignación de datos

MapForce valida las asignaciones automática cuando se hace clic en el panel *Resultados* para obtener una vista previa de los resultados de la transformación. También puede validar la asignación de forma explícita para identificar y corregir errores antes de ejecutar la asignación. Tenga en cuenta que mientras se ejecuta una asignación pueden darse errores de tiempo de ejecución o advertencias, dependiendo del tipo de datos procesado (p. ej. cuando se sobrescriben valores asignado a atributos).

Hay dos maneras de validar una asignación de forma explícita:

- Haciendo clic en el comando de menú **Archivo | Validar asignación**.
- Haciendo clic en el botón **Validar**  de la barra de herramientas.

El resultado de la validación aparece en la ventana Mensajes:



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 con 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 generarse 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

Icono	Significado
	XQuery.

Para ver resaltar en el área de asignación el componente o la estructura que dio lugar al mensaje de información, advertencia o error basta con hacer clic en el texto subrayado del mensaje.

Cuando la asignación contiene componentes que transforman datos (p. ej. funciones o variables), la validación funciona de la siguiente manera:

- 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 documento de salida.

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.

La ventana Mensajes también ofrece botones para:

- Filtrar los mensajes por tipo (p. ej. para ver solamente errores o advertencias).
- Navegar hacia arriba o hacia abajo por los mensajes.
- Copiar el texto del mensaje en el portapapeles.
- Buscar texto concreto en la ventana.
- Borrar el contenido de la ventana Mensajes.

Para más información consulte el apartado Interfaz del usuario.

4.1.5 Validar resultados de la asignación

Al hacer clic en el panel *Resultados*, aparece una vista previa del resultado de la asignación. Este resultado se puede validar usando el esquema que tenga asociado. Por ejemplo, si la transformación genera un archivo XML, puede validarse con el esquema XML asociado.

Cuando trabaje con archivos XML puede especificar el esquema asociado al archivo de instancia en el campo *Agregar referencia de esquema/DTD* del cuadro de diálogo "Configuración del componente" (*véase Configuración de componentes XML*). La ruta de acceso especifica la ubicación del archivo de esquema al que hace referencia el archivo XML de salida. Esto permite poder validar el archivo de instancia de salida cuando se ejecute la asignación. En el campo *Agregar referencia de esquema/DTD* puede introducir una dirección HTTP o una ruta de acceso absoluta o relativa. Si no marca la casilla *Agregar referencia de esquema/DTD,* no se podrá validar el archivo de salida. Por el contrario, si marca la casilla pero deja el cuadro de texto vacío, el nombre de archivo del esquema se genera en el resultado la validación se lleva a cabo con este nombre de esquema.

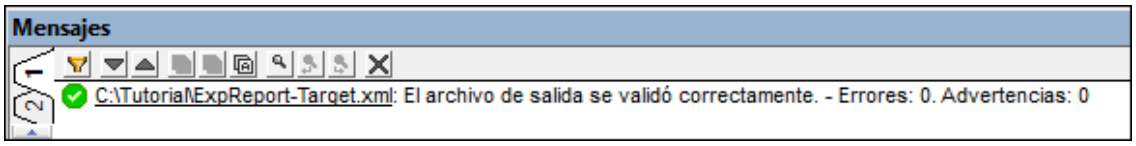
Para validar el resultado de la asignación tiene dos opciones:

- Hacer clic en el botón **Validar archivo de salida** de la barra de herramientas.

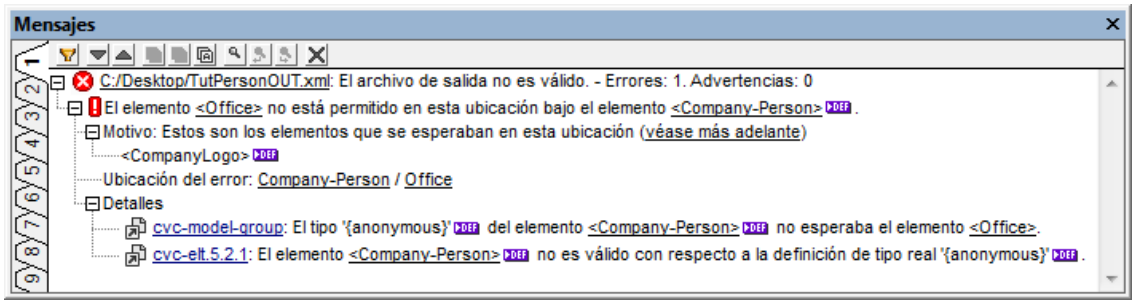
- Hacer clic en el comando de menú **Resultados | Validar el archivo de salida**.

Note: 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.

El resultado de la validación aparece en la ventana Mensajes. Por ejemplo:



Si se produce un error de validación, la ventana Mensajes ofrece información al respecto:



Los mensajes de validación contienen hipervínculos que dirigen a información más detallada:

- Haga clic en la ruta de acceso del archivo para abrir el resultado de la transformación en el panel *Resultados* de MapForce.
- Haga clic en los enlaces de tipo `<nombreElemento>` para resaltar el elemento en el panel *Resultados*.
- Haga clic en el icono para abrir la definición del elemento en Altova XMLSpy (si está instalado).
- Haga clic en los hipervínculos de la sección *Detalles* (p. ej. `cvc-model-group`) para abrir la descripción de la regla de validación correspondiente en el sitio web http://www.w3.org/

.

4.1.6 Vista previa de resultados

En MapForce puede consultar una vista previa de los resultados de la asignación 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 intentar 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*.

Una vez disponibles en el panel *Resultados*, los datos se pueden validar y guardar (*véase Validar resultados de la asignación*). También puede buscar texto dentro del archivo de salida con el comando **Buscar** (**Ctrl+F**). Consulte también el apartado *Búsquedas en la vista Texto*.

Los mensajes de error, advertencia e información relacionados con la ejecución de la asignación aparecen en la ventana Mensajes (*véase Interfaz del usuario*).

Para generar la vista previa de resultados:

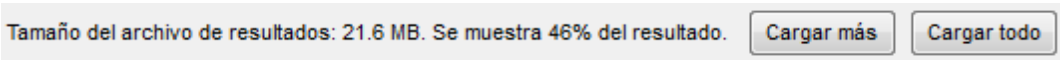
- Haga clic en el panel *Resultados* situado en la parte inferior de la ventana de asignación. MapForce usa el lenguaje de transformación seleccionado en la barra de herramientas **Lenguaje** para ejecutar la asignación y rellena el panel *Resultados* con el resultado.

Puede guardar el resultado de la transformación de dos maneras diferentes:

- 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.

Vista previa parcial de los resultados

A la hora de generar una vista 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. Al hacer clic en este botón se anexa la siguiente parte del archivo a la vista previa actual y así sucesivamente.



Nota: el botón **Pretty-print** se habilita cuando está cargado todo el archivo en el panel *Resultados*.

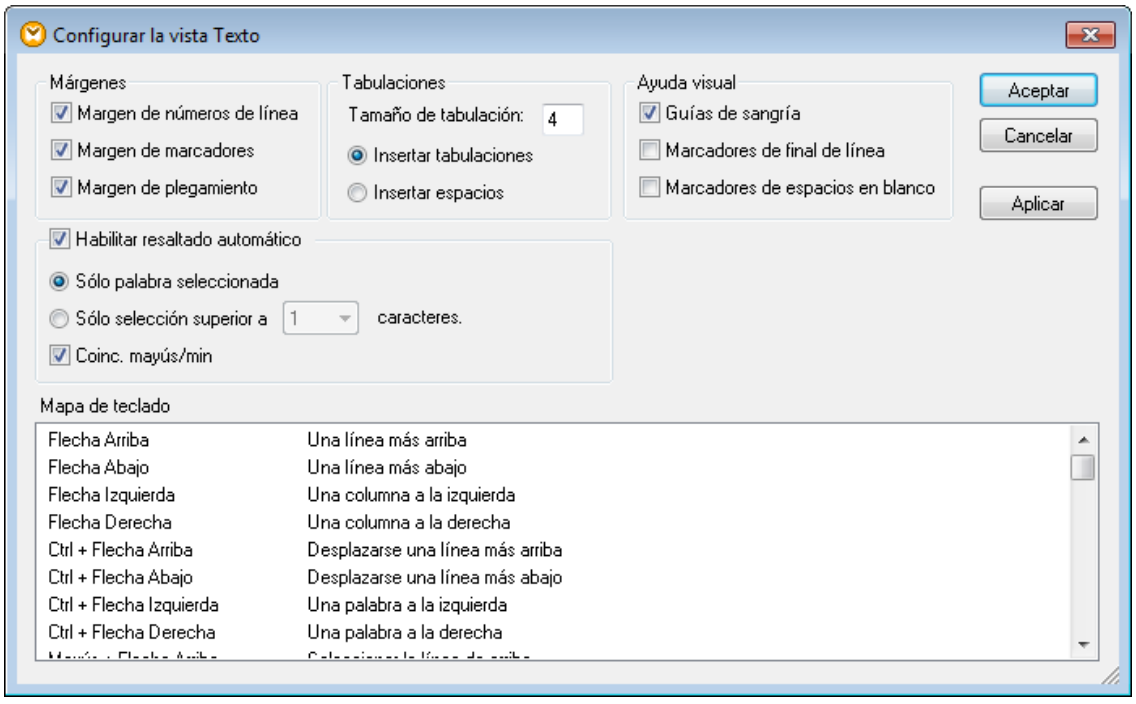
La vista previa también se puede configurar en la pestaña *Generales* del cuadro de diálogo "Opciones" (*véase Cambiar opciones de MapForce*).

4.1.7 Características de la vista Texto

Los paneles *Resultados* y *XSLT* ofrecen varias funciones visuales para facilitar la consulta y edición del texto. Estas son las características de la vista Texto:

- Números de línea
- Color de sintaxis
- Marcadores
- Plegamiento de código
- Guías de sangría
- Marcadores de final de línea y marcadores de espacios en blanco
- Zoom
- Formato pretty-print
- Ajuste automático de línea
- Resaltado de texto

Estas características se pueden habilitar, deshabilitar y personalizar en el cuadro de diálogo "Configurar la vista Texto". La configuración elegida en este cuadro de diálogo se aplica a toda la aplicación y no solo al documento activo.

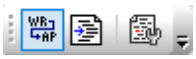


Cuadro de diálogo "Configurar la vista Texto"

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.



Barra de herramientas Vista Texto

Para ver la lista de teclas de acceso rápido consulte el panel *Mapa de teclado* situado en la parte inferior del cuadro de diálogo "Configurar la vista Texto" (*imagen anterior*).

Números de líneas

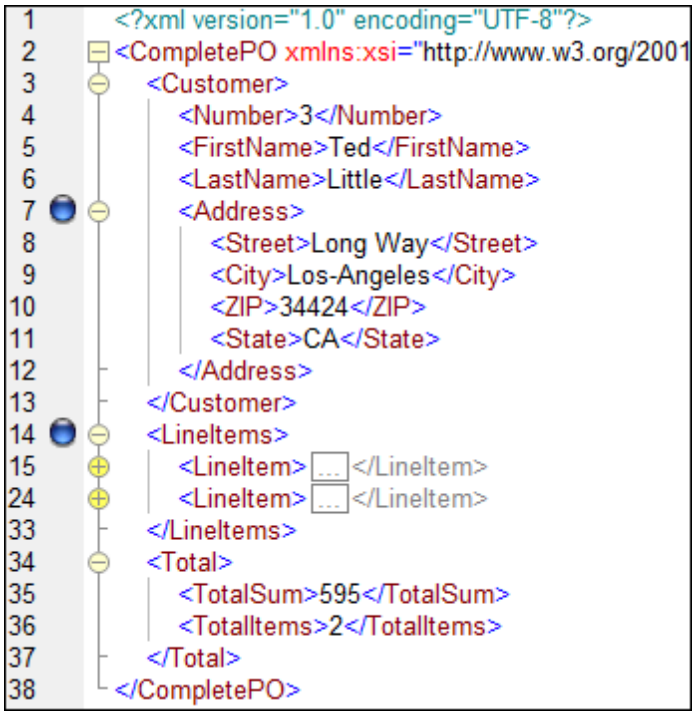
Los números de línea aparecen en el margen de números de línea (*imagen anterior*), que se puede activar o desactivar en el cuadro de diálogo "Configurar la vista Texto". Cuando se contrae una sección de texto, los números de línea del texto contraído también se ocultan.

Color de sintaxis

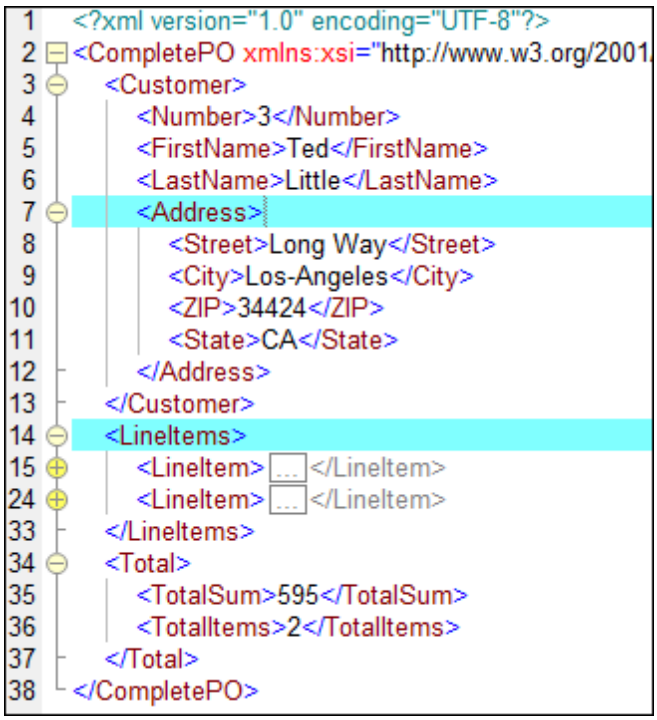
El color de sintaxis se aplica de acuerdo al valor semántico del texto. Por ejemplo, en los documentos XML, dependiendo de si el nodo XML es un elemento, un atributo, contenido, una sección CDATA, un comentario o una instrucción de procesamiento, el nombre del nodo (y en ocasiones el contenido del nodo) tendrá un color diferente.

Marcadores

Las líneas del documento se pueden marcar para realizar consultas rápidas más adelante. Si el margen de marcadores está activado, los marcadores aparecen en el margen de marcadores.



De lo contrario, las líneas marcadas se resaltan en color cian.



El margen de marcadores se puede activar o desactivar en el cuadro de diálogo "Configurar la vista Texto".

Puede navegar por los marcadores y editarlos con estos comandos:

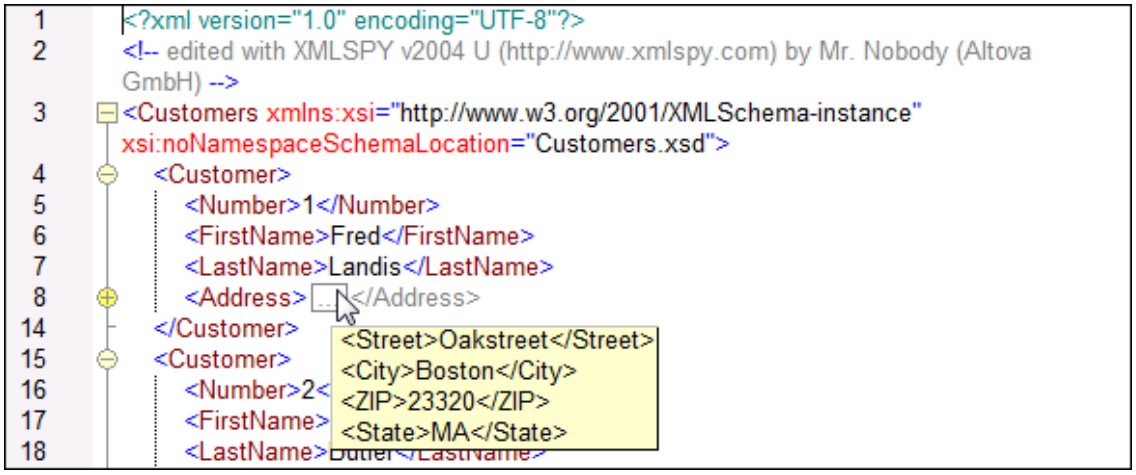
- **Insertar o quitar marcador (Ctrl + F2)**
- **Ir al siguiente marcador (F2)**
- **Ir al marcador anterior (Mayús + F2)**
- **Eliminar todos los marcadores (Ctrl + Mayús + F2)**

Todos 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* y *XSLT*.

Plegamiento de código

La característica plegamiento de código permite expandir y contraer nodos en documentos XML, XQuery, JSON y CSS. Los nodos que se pueden expandir/contraer se marcan en el margen de plegamiento de código con el signo **+/-** (*imagen siguiente*). El margen puede activarse o desactivarse en el cuadro de diálogo "Configurar la vista Texto". Cuando está contraído, el nodo se marca con puntos suspensivos. Si pasamos el puntero por encima de los puntos suspensivos, aparece en pantalla un cuadro emergente con el contenido del nodo contraído (ver marca azul en

la imagen). Si el contenido del nodo es demasiado largo para el cuadro emergente, esto se indica con puntos suspensivos también.



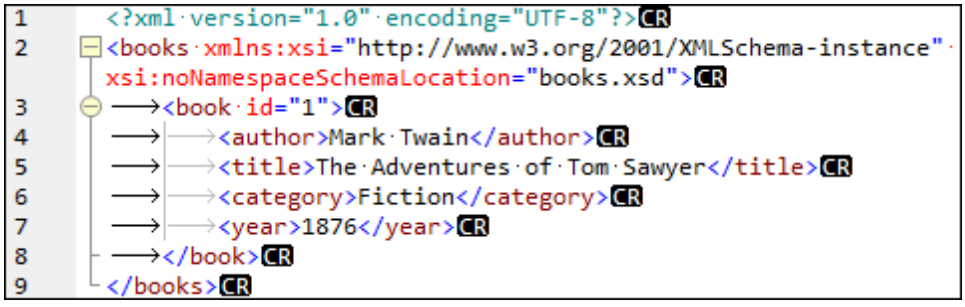
Guías de sangría

Las guías de sangría son líneas de puntos 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".

Nota: las opciones *Insertar tabulaciones* e *Insertar espacios* del cuadro de diálogo "Configurar la vista Texto" 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 (**EOL**) y los marcadores de espacios en blanco se pueden activar o desactivar en el cuadro de diálogo "Configurar la vista Texto". La captura de pantalla que aparece a continuación incluye estos marcadores. Cada punto representa un espacio en blanco, cada flecha representa un carácter de tabulación y CR representa un retorno de carro.



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 se desplaza cada nodo secundario de su nodo principal insertando cuatro caracteres de espaciado. Esta configuración se puede personalizar en el cuadro de diálogo "Configuración de 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

Para activar el ajuste automático de línea en el documento activo haga clic en el botón **Ajuste automático de línea** de la barra de herramientas.

Resaltado de texto

Si se habilita la función de resaltado de texto en el cuadro de diálogo "Configurar la vista Texto", la aplicación resalta en el documento todas las coincidencias del texto seleccionado por el usuario. La selección se resalta en azul pálido y las coincidencias se resaltan en naranja pálido. La selección y sus coincidencias aparecen marcadas en la barra de desplazamiento por medio de marcadores grises cuadrados. Observe que la barra de desplazamiento también indica la posición actual del cursor por medio de un marcador azul.

Para habilitar la función de resaltado basta con marcar la casilla *Habilitar resaltado automático* en el cuadro de diálogo "Configurar la vista Texto". 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. En este caso, si selecciona un solo carácter del documento, no se encontrarán coincidencias. Pero si selecciona dos o más caracteres, sí se encontrarán coincidencias. Por ejemplo, si selecciona `t`, la aplicación no encontrará coincidencias. Pero si selecciona `ty`, la aplicación mostrará todas las coincidencias que contengan `ty`; si selecciona `typ`, la aplicación mostrará todas las coincidencias que contengan `typ` y así sucesivamente.

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.

4.1.8 Búsquedas en la vista Texto

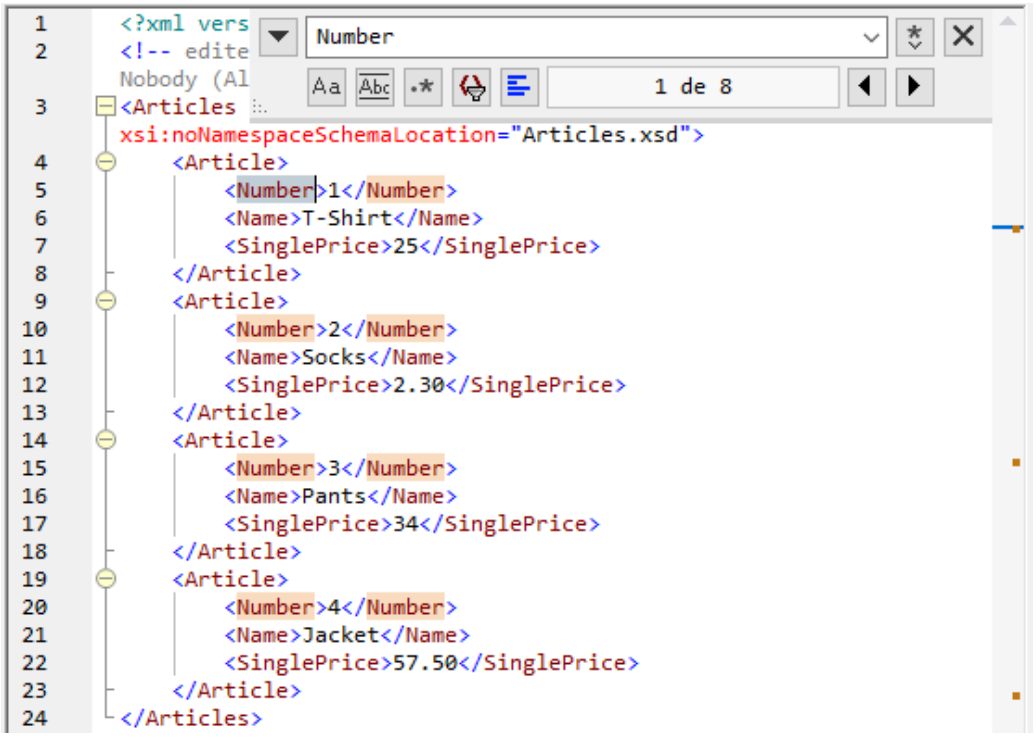
Puede realizar búsquedas en el texto de los paneles *Resultados* y *XSLT* con ayuda de numerosas características gráficas.

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.

- Introduzca la cadena de texto que se debe buscar o seleccione una cadena de texto

buscada en 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 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. Por ejemplo, **2 de 4** indica que el segundo resultado (de un total de cuatro) está seleccionado.
- Puede recorrer los resultados en ambos sentidos con los botones **Anterior** **(Mayús +F3)** y **Siguiente** **(F3)** situados en la esquina inferior derecha.



Para cerrar el cuadro de diálogo "Buscar" haga clic en el botón **Cerrar** situado en la esquina superior derecha o pulse **Esc**.

Debe tener en cuenta estas características de la función de búsqueda y reemplazo:

- El cuadro de diálogo "Buscar" no es modal. Esto significa que puede dejarlo abierto mientras utiliza la vista Texto.
- Si tiene texto seleccionado antes de abrir el cuadro de diálogo, el texto seleccionado se inserta automáticamente en el campo del término de búsqueda.
- 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**  .

- Después de cerrar el cuadro de diálogo "Buscar" puede repetir la búsqueda actual con solo pulsar **F3** (búsqueda hacia adelante) o **Mayús+F3** (búsqueda hacia atrás). El cuadro de diálogo aparece al pulsar estas teclas.

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:

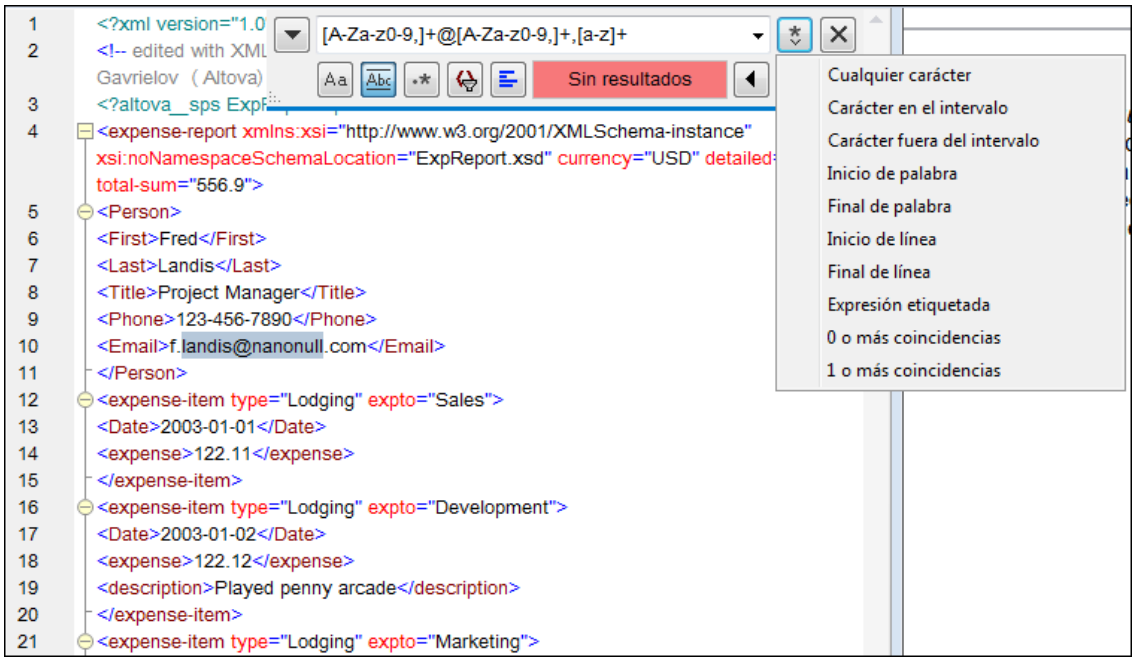
Opción	Icono	Descripción
Coinc. mayús/ min		La búsqueda tiene en cuenta las mayúsculas y minúsculas a la hora de realizar la búsqueda (`Address` no es lo mismo que `address`).
Sólo palabras completas		Solo se consideran coincidencias las palabras completas. Por ejemplo, si activa esta opción y busca el término `fit`, la aplicación devuelve como resultado la palabra `fit`, pero no `fitness`.
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 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 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.



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). Las expresiones regulares son similares a las subexpresiones (grupos indizados) de la versión .NET de las expresiones regulares. 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*).
`\<`	Inicio de palabra.

\>	Final de palabra.
\	Escapes the character following the backslash. In other words, the expression \x allows you to use the character x literally. For example, \[would be interpreted as [and not as the start of a character set. 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

4.1.9 Vista previa del código XSLT

Puede consultar una vista previa del código XSLT generado con MapForce seleccionando **XSLT 1.0** o **XSLT 2.0** como lenguaje de transformación (*véase* *Seleccionar el lenguaje de transformación*).

Para obtener la vista previa del código XSLT generado:

- Si se trata de código XSLT 1.0, haga clic en el botón **XSLT** situado en la parte inferior de la ventana de asignación.
- Si se trata de código XSLT 2.0, haga clic en el botón **XSLT2** situado en la parte inferior de la ventana de asignación.

Nota: los paneles *XSLT* y *XSLT2* de la ventana de asignación solamente se habilitan cuando el lenguaje de transformación seleccionado es **XSLT** o **XSLT2** respectivamente.

4.1.10 Generar código XSLT

Para generar código XSLT:

1. Seleccione el comando de menú **Archivo | Generar código en | XSLT 1.0 (XSLT 2.0)**.
2. Seleccione la carpeta donde desea guardar el archivo XSLT generado y haga clic en **Aceptar**. MapForce genera el código y muestra el resultado de la operación en la ventana Mensajes.

El nombre del archivo .xslt que se genera tiene el formato **<A>MapTo<B>.xslt**:

- "<A>" es el valor del campo *Nombre de la aplicación* del cuadro de diálogo "Configurar asignación" (*véase Cambiar configuración de la asignación*).
- "<B>" es el nombre del componente de destino. Para cambiar este valor abra la configuración del componente de destino y edite el valor del campo *Nombre del componente* (*véase Cambiar configuración de los componentes*).

La carpeta donde se guarda el archivo .xslt también contiene un archivo por lotes llamado **DoTransform.bat** que se puede ejecutar con RaptorXML Server para transformar los datos (*véase Automatización con RaptorXML Server*).

Para ejecutar la transformación con RaptorXML Server:

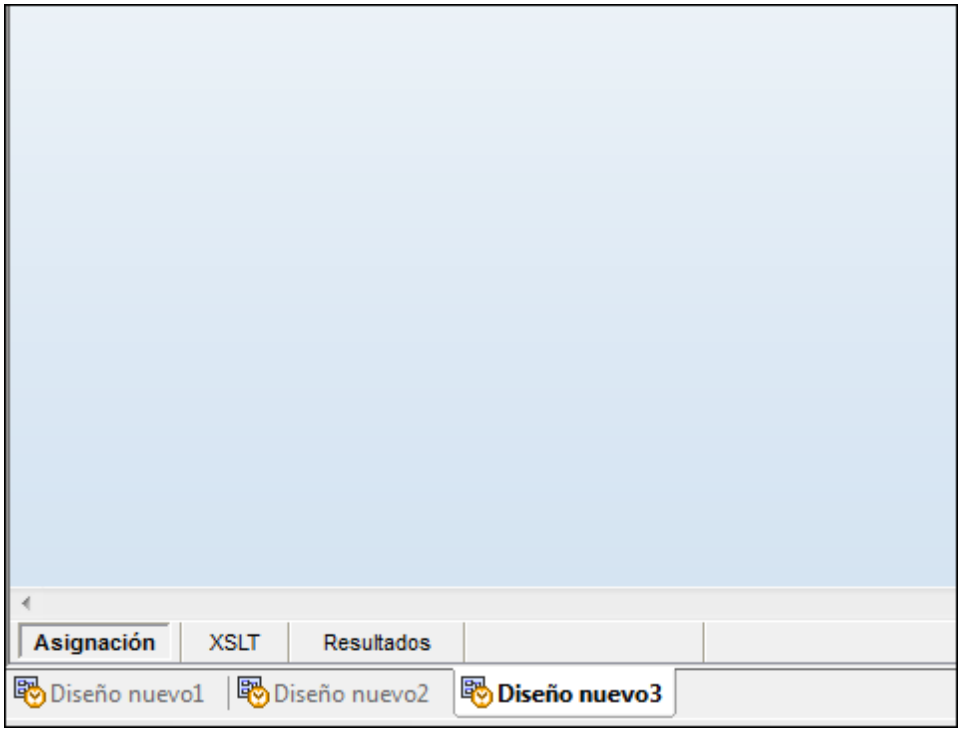
1. Descargue e instale el software servidor RaptorXML Server desde la página de descargas de Altova (https://www.altova.com/es/download-trial-server.html).
2. Inicie el archivo de procesamiento por lotes **DoTransform.bat** situado en la carpeta de salida designada con anterioridad.

Recuerde que a veces es necesario agregar la carpeta de instalación de RaptorXML a la variable **path** de las variables de entorno. Para más información consulte la documentación de RaptorXML en el sitio web de Altova (https://www.altova.com/es/documentation.html).

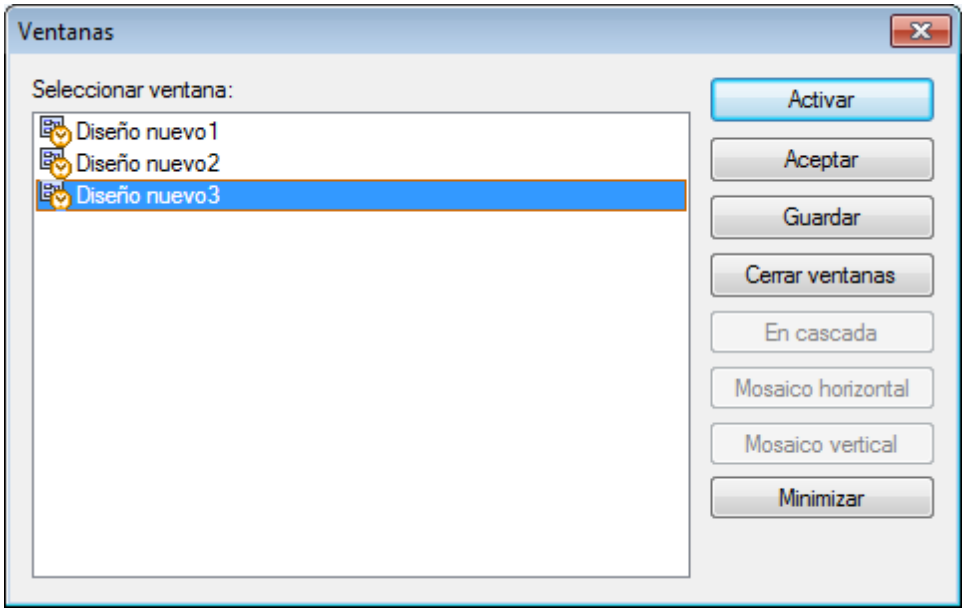
4.1.11 Trabajar con varias ventanas de asignación

MapForce utiliza la especificación de interfaz de múltiples documentos MDI. Cada archivo de asignación que se abre en MapForce tiene una ventana distinta. Esto permite trabajar con varias ventanas de asignación y organizarlas o ajustar su tamaño dentro de la ventana de la aplicación. También puede organizar las ventanas abiertas en mosaico horizontal o vertical y en cascada.

Cuando hay varias asignaciones abiertas en MapForce puede navegar por ellas haciendo clic en las pestañas que aparecen en la parte inferior del panel de asignación.



Además, el menú **Ventanas** ofrece comandos para gestionar las ventanas de las asignaciones y el comando **Ventanas | Ventanas...** abre el cuadro de diálogo del mismo hombre donde puede realizar operaciones como guardar, cerrar o minimizar ventanas.



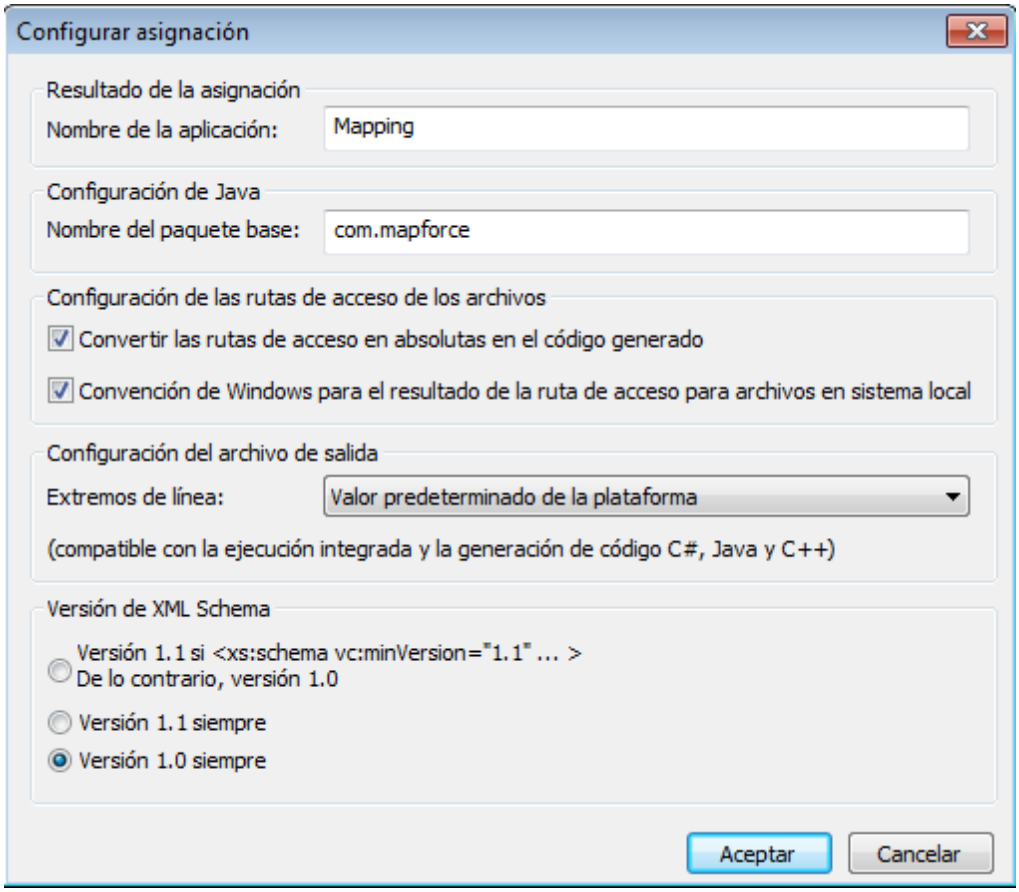
Para seleccionar varias ventanas en el cuadro de diálogo "Ventanas" haga clic en las entradas pertinentes mientras pulsa la tecla **Ctrl**.

4.1.12 Cambiar configuración de la asignación

Puede cambiar la configuración del diseño de asignación activo desde el cuadro de diálogo "Configurar asignación". Las opciones de configuración definidas en este cuadro de diálogo se almacenan en el archivo `*.mfd`.

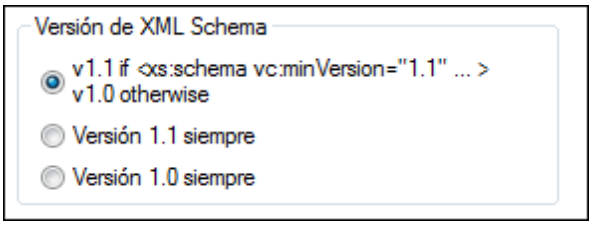
Para abrir el cuadro de diálogo "Configurar asignación":

- Haga clic en el comando de menú **Archivo | Configurar asignación**.



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

Nombre de la aplicación	Define el prefijo de nombre del archivo XSLT1.0/2.0 para los archivos de transformación generados.
Convertir las rutas de acceso en absolutas en el código generado	Define si las rutas de acceso de los archivos deben ser relativas o absolutas en el código de programa generado. Consulte Rutas de acceso en el código generado para obtener más información.
Convención de Windows para	Marque esta casilla para que MapForce siga la convención

el resultado de la ruta de acceso para archivos en sistema local	de Windows para rutas de acceso. Cuando genere código XSLT2 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 precesamiento
Versión de XML Schema	En este grupo de opciones puede indicar qué versión de XML Schema se usa en el archivo de asignación. Aquí puede indicar si prefiere **cargar** siempre los esquemas conforme a la versión 1.0 o conforme a la versión 1.1. 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. **Nota:** 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.

4.2 Trabajar con componentes

Los componentes son los elementos centrales de los diseños de asignación de datos en MapForce. Por lo general, el término *componente* se utiliza para denominar cualquier objeto que actúe como origen o destino de datos o que represente los datos en la asignación en una fase de procesamiento intermedio.

Hay dos categorías de componentes: componentes de estructura y componentes de transformación.

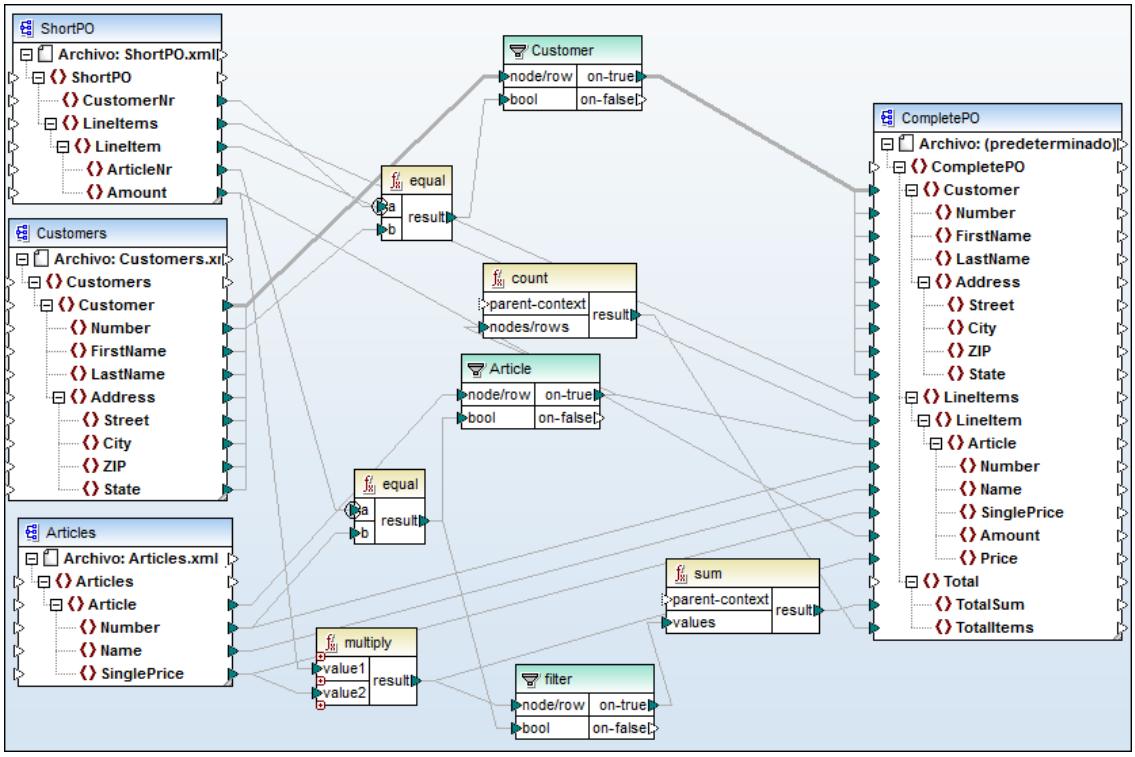
Los **componentes de estructura** representan la estructura abstracta o el esquema de los datos. Por ejemplo, cuando se añade un archivo XML al área de asignación (con el comando de menú **Insertar | Archivo o esquema XML**), el archivo se convierte en un componente de la asignación. Un **elemento** es la unidad de asignación de menor nivel (por ejemplo, un atributo de un archivo XML o un elemento de tipo simple). Y una **secuencia** es una colección de elementos.

Los **componentes de transformación** transforman datos (p. ej. las funciones) o prestan ayuda durante la transformación (p. ej. las constantes y variables). Para más información consulte la sección Diseño de asignaciones.

Con la ayuda de los componentes de estructura podrá leer datos de archivos y otros orígenes de datos, escribir datos en archivos y otros orígenes o almacenar datos en una fase intermedia del proceso de asignación (p. ej. para obtener una vista previa de los datos). Por tanto, podemos hablar de tres tipos de componentes de estructura:

- **Origen:** un componente se declara como componente de origen al colocarlo en el lado izquierdo del área de asignación (dando a MapForce la orden de leer datos de este componente).
- **Destino:** un componente se declara como componente de destino al colocarlo en el lado derecho del área de asignación (dando a MapForce la orden de escribir datos en este componente).
- **Paso a través:** este tipo especial de componente hace las veces de origen y destino a la vez (para más información consulte el apartado Asignaciones en cadena / componentes de paso a través).

En el área de asignación los componentes toman la forma de rectángulos. Por ejemplo, en esta imagen la asignación está compuesta por tres componentes de origen, un componente XML de destino y varios componentes de transformación (funciones y filtros) por los que se pasan los datos antes de escribirse en el destino.



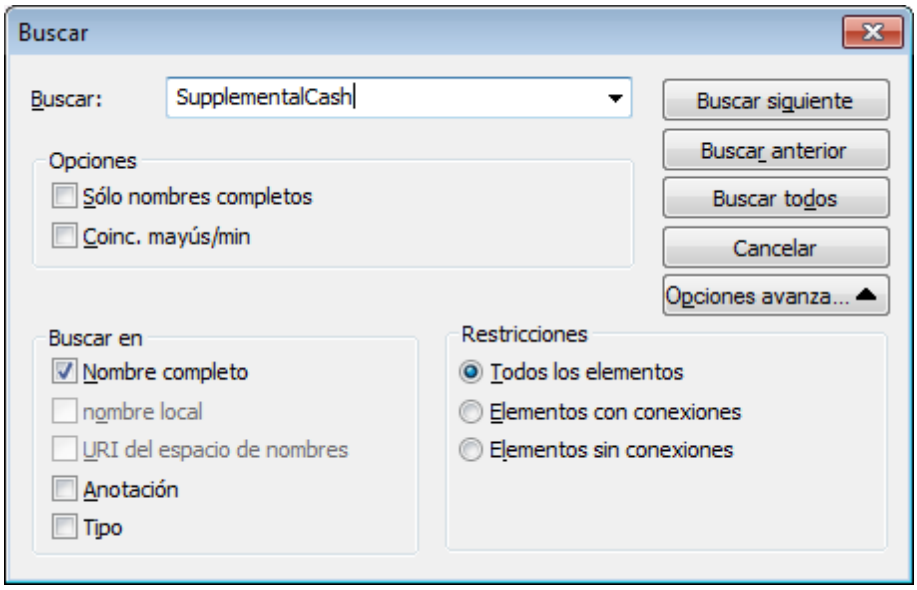
CompletePO.mfd

Este ejemplo de asignación está disponible en esta ruta de acceso: `<Documentos>\Altova \MapForce2018\MapForceExamples\CompletePO.mfd`.

4.2.1 Búsquedas dentro de los componentes

Para buscar un elemento o nodo determinado de un componente:

1. Haga clic en el componente donde se debe buscar y pulse **Ctrl+F**.
2. Introduzca el término de búsqueda y haga clic en **Buscar siguiente**.

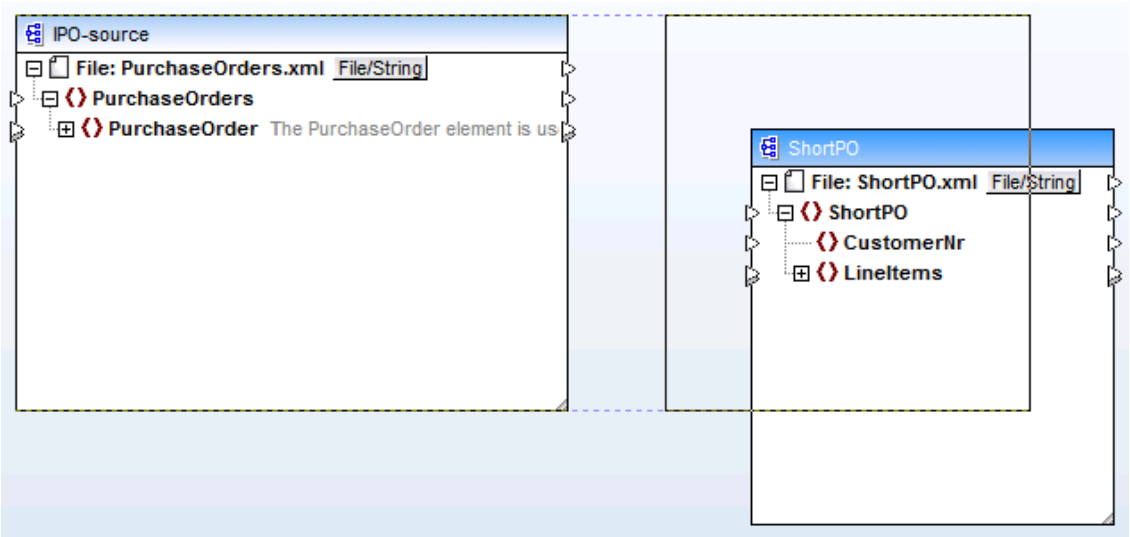


Haga clic en el botón **Opciones avanzadas** para definir en qué tipo de elementos (nodos) se debe realizar la búsqueda y restringir las opciones de búsqueda a determinadas conexiones solamente.

4.2.2 Alinear componentes

Mientras el usuario mueve componentes por el área de asignación MapForce muestra líneas de guía para ayudarle a alinear los componentes del diseño correctamente.

Por ejemplo, en la imagen siguiente el usuario sube el componente situado a la derecha un poco más arriba en el área de asignación. Las líneas de guía indican que el componente que se está moviendo puede alinearse con el componente situado a la izquierda.



Líneas de guía para la alineación de componentes.

Para habilitar o deshabilitar las líneas de guía:

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

4.2.3 Cambiar configuración de los componentes

Tras añadir un componente al área de asignación podrá configurarlo desde el cuadro de diálogo "Configuración del componente". Este cuadro de diálogo se puede abrir de varias maneras:

- Seleccionando el componente y haciendo clic en el comando de menú **Componente | Propiedades**.
- Haciendo doble clic en el título del componente.
- Haciendo clic con el botón derecho en el título del componente y seleccionando **Propiedades** en el menú contextual.

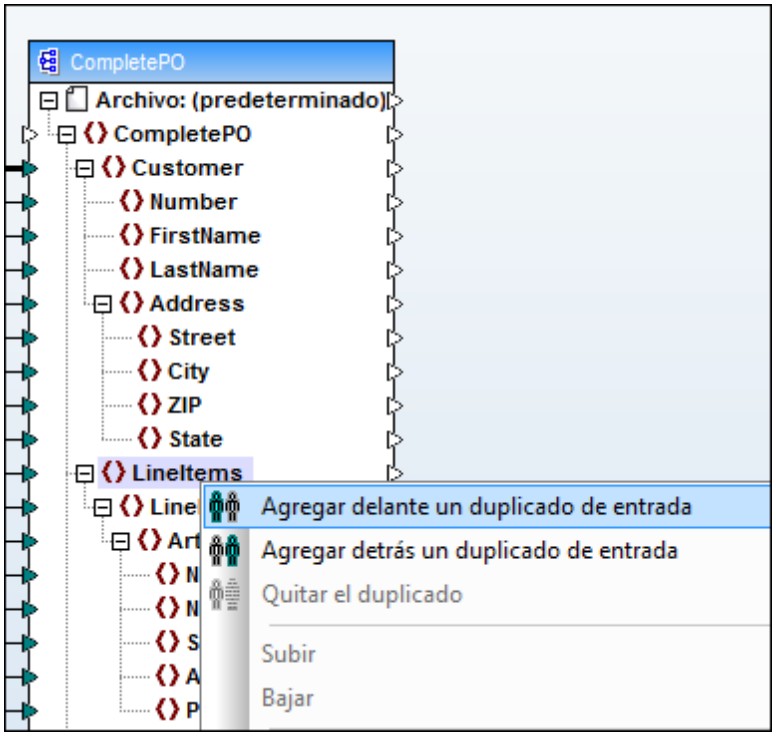
En el apartado Configuración de componentes XML se describen todas las opciones de configuración.

En los componentes basados en archivos, como los componentes XML por ejemplo, junto al nodo raíz aparece el botón Archivo. Este botón sirve para especificar opciones avanzadas cuando se quiera procesar o generar varios archivos de salida en la misma asignación (*véase Procesar varios archivos de entrada o salida simultáneamente*).

4.2.4 Duplicar entradas

A veces es necesario configurar un componente para que acepte datos de varios orígenes de datos a la vez. Por ejemplo, imagine que necesita convertir datos de dos esquemas XML diferentes y pasarlos a un solo esquema. Para que el esquema de destino acepte los datos de ambos esquemas de origen, lo más fácil es crear duplicados de uno de los elementos de entrada dentro del componente. Los duplicados de entrada están diseñados solamente para los componentes de destino y pueden crearse tantos duplicados de entrada como sean necesarios.

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.



En la imagen anterior puede ver que el elemento `LineItem` se va a duplicar para poder asignar datos de otro origen de datos más.

Tras crear el duplicado de entrada se pueden realizar conexiones tanto con la entrada original como con el duplicado de entrada. Por ejemplo, puede copiar datos del origen A a la entrada original y datos del origen B al duplicado de entrada.

Nota: no está permitido crear duplicados de atributos XML porque daría lugar a una instancia XML no válida. La duplicación de elementos XML de entrada está permitida independientemente del valor que tenga el atributo `maxOccurs` en el esquema. Esto se permite porque el esquema puede cambiar y porque los datos de origen pueden ser opcionales. Por ejemplo, una asignación podría generar un solo elemento XML incluso si la entrada está duplicada en el diseño de asignación.

Para ver un ejemplo paso a paso consulte el apartado Asignar varios orígenes de datos a un destino.

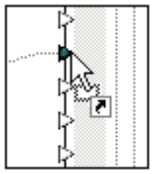
4.3 Trabajar con conexiones

Una asignación de datos tiene como último objetivo transformar datos de un formato a otro o transformar su estructura a otra distinta. Tomemos una asignación muy básica como ejemplo: el usuario añade al área de asignación los componentes que representan sus datos de origen y de destino (p. ej. un esquema XML de origen y uno de destino) y después dibuja las conexiones de asignación entre las dos estructuras. Por tanto, una conexión no es más que la representación visual de la asignación de datos entre el componente de origen y el de destino.

Los componentes tienen entradas y salidas que aparecen representados en la asignación en forma de triángulos y reciben el nombre de *conectores*. Los conectores de entrada están situados a la izquierda de los elementos para los que se puede dibujar una conexión. Los conectores de salida, por otro lado, están situados a la derecha de los elementos para los que se puede dibujar una conexión.

Para dibujar una conexión entre dos elementos:

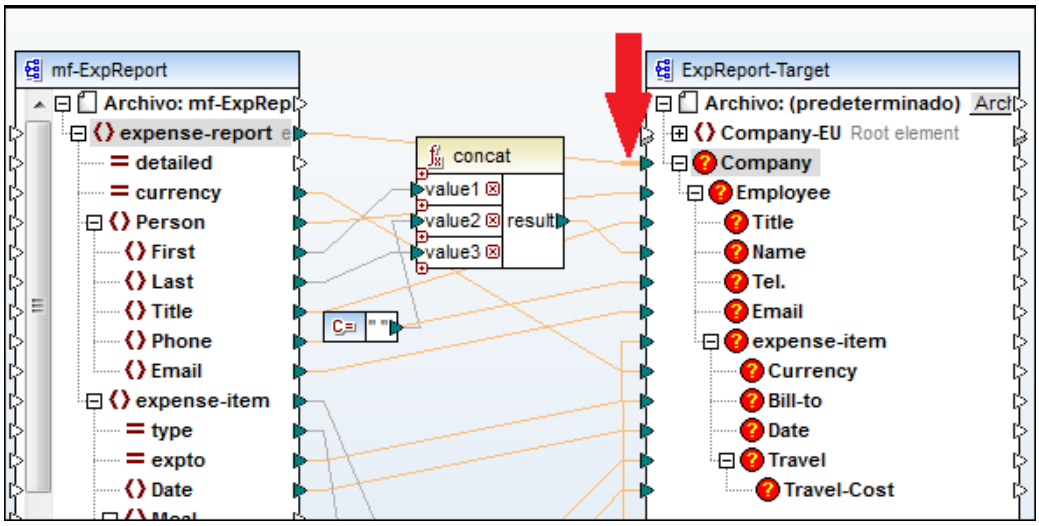
- Haga clic en el conector de salida de un elemento de origen y arrastre el puntero hasta el elemento de destino. Cuando la acción de colocar esté permitida, aparece junto al cursor aparece un icono de enlace.



Cada conector de entrada acepta un máximo de una conexión de entrada. Si intenta agregar otra conexión a la misma entrada, aparece un cuadro de mensaje preguntando si desea reemplazar la conexión con una nueva o crear un duplicado del elemento de entrada. Cada conector de salida, por el contrario, puede tener varias conexiones con diferentes entradas.

Para mover una conexión a otro elemento:

- Haga clic en el extremo de la conexión (es decir, la sección recta situada junto al destino) y arrástrelo hasta el nuevo destino.

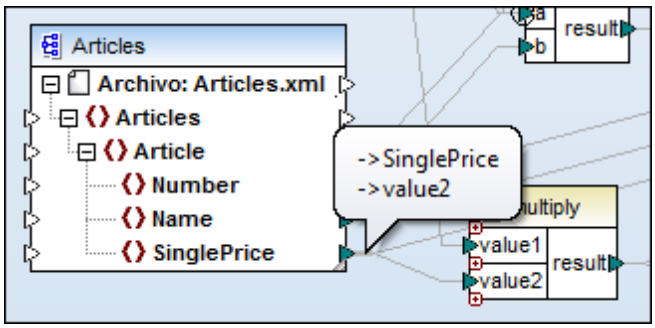


Para copiar una conexión con otro elemento:

- Haga clic en el extremo de la conexión (es decir, la sección recta situada junto al destino) y arrástrelo hasta el nuevo destino mientras pulsa la tecla **Ctrl**.

Para ver el elemento situado en el otro extremo de una conexión:

- Coloque el puntero sobre la sección recta situada junto a un extremo de la conexión. Aparece un mensaje de información rápida con el nombre de los elementos situados en el otro extremo de la conexión. Si se definieron varias conexiones desde la misma salida, la información rápida ofrece un máximo de diez nombres de elemento. En la imagen siguiente, por ejemplo, los dos elementos de destino de la función de multiplicación son `SinglePrice` y `value2`.



Para cambiar la configuración de la conexión tiene tres opciones:

- En el menú **Conexión** haga clic en el comando **Propiedades** (este comando solamente se habilita si tiene seleccionada una conexión).
- Haga doble clic en la conexión.
- Haga clic con el botón derecho en la conexión y después elija **Propiedades** en el menú contextual.

Consulte el apartado Configuración de las conexiones para obtener más información.

Para eliminar una conexión tiene dos opciones:

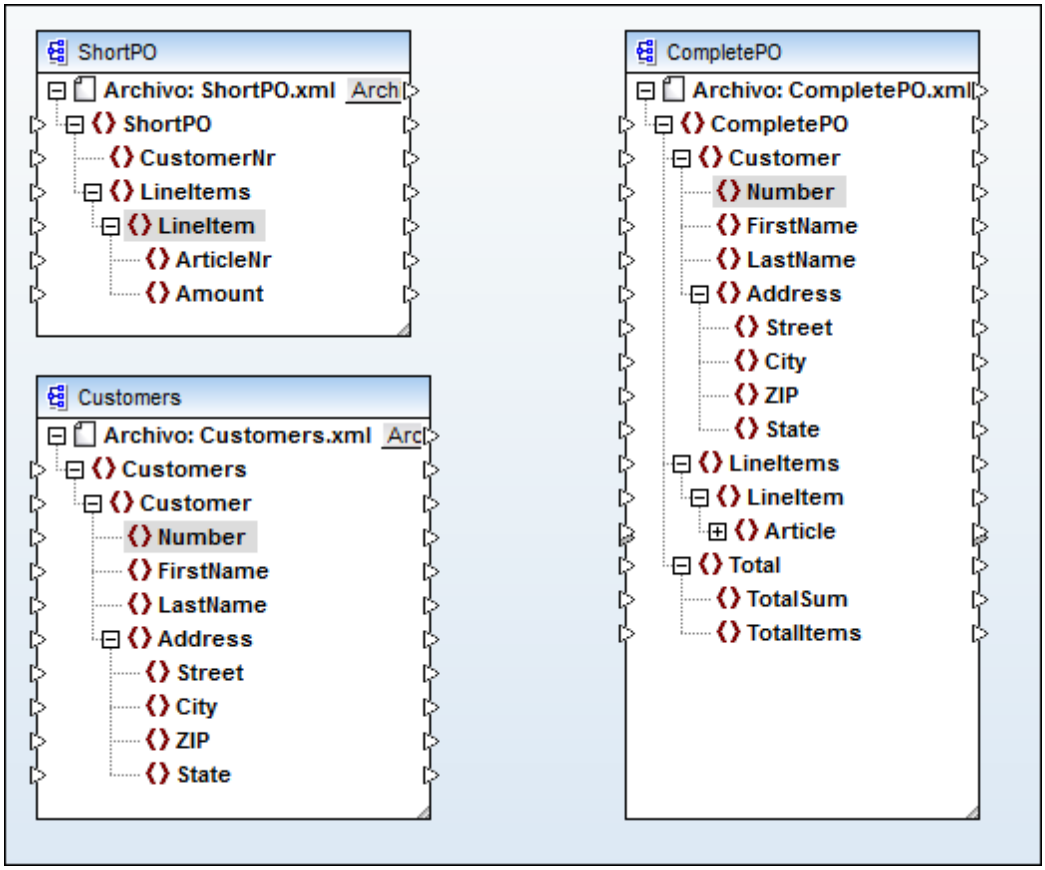
- Haga clic en la conexión y pulse la tecla **Supr**.
- Haga clic con el botón derecho en la conexión y elija **Eliminar** en el menú contextual.

4.3.1 Nota sobre entradas obligatorias

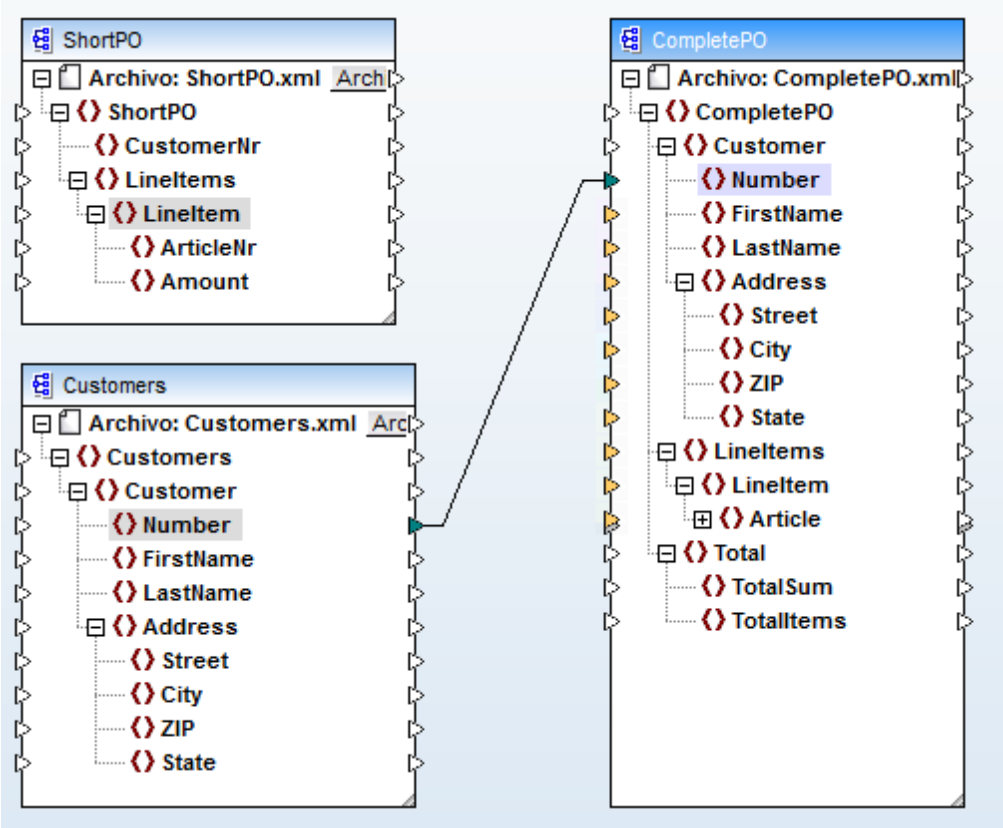
MapForce resalta en color naranja las entradas obligatorias de los componentes de destino:

- En los componentes XML y EDI son entradas obligatorias los elementos cuyo parámetro `minOccurs` tiene un valor igual o mayor que 1.
- En las bases de datos son entradas obligatorias los campos definidos como "not null".
- Todos los nodos de las llamadas y respuestas WSDL.
- Todos los nodos XBRL definidos como obligatorios.
- En las funciones, una vez creada una asignación con un parámetro, MapForce resalta los parámetros obligatorios para señalar que es necesario realizar una conexión. Por ejemplo: una vez creada la asignación con los parámetros de entrada de un filtro, el otro parámetro se resalta automáticamente.
- En las hojas de cálculo MS Excel son entradas obligatorias todos los nombres de hoja de cálculo.

Ejemplo (asignación de datos CompletePO.mfd, disponible en la carpeta . . . `\MapForceExamples`)
Cuando creamos una asignación como la de este archivo de ejemplo, los archivos de esquema XML insertados tienen este aspecto:



Después conectamos el elemento `Number` del componente **`Customers`** con el elemento `Number` del componente **`CompletePO`**. Nada más crearse la conexión, MapForce resalta los elementos/nodos obligatorios del componente **`CompletePO`**. Observe que también se resalta el nodo/icono contraído `Article`.



4.3.2 Opciones de presentación de las conexiones

Puede elegir qué tipo de conexiones aparecen en la ventana de asignación.

El botón **Mostrar los conectores del componente seleccionado** permite alternar entre:

- todos los conectores de la asignación, que aparecen en negro y
- todos los conectores relacionados con el componente seleccionado, que aparecen en negro. Los demás conectores aparecen atenuados.

El botón **Mostrar los conectores desde su origen a su destino** permite alternar entre:

- los conectores que están conectados con el componente seleccionado **directamente** y
- los conectores que están unidos al componente seleccionado, que parten del origen y terminan en los componentes de destino.

4.3.3 Crear anotaciones en las conexiones

MapForce ofrece la posibilidad de etiquetar conexiones para poder comentar la asignación en detalle. Esta característica está disponible para **todos los tipos de conexiones**.

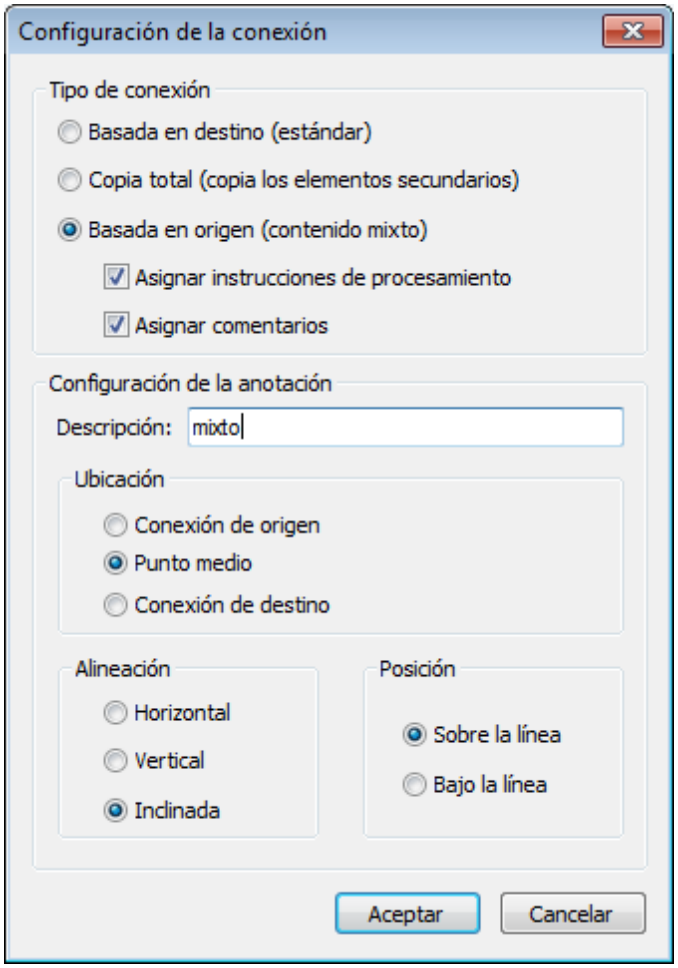
Para anotar una conexión:

1. Haga clic con el botón derecho en la conexión y elija **Propiedades** en el menú contextual.
2. En el campo *Descripción* introduzca un nombre para la conexión seleccionada. Al introducir el texto se habilitan las opciones del grupo *Configuración de la anotación*.
3. Defina la *ubicación*, la *alineación* y la *posición* de la etiqueta.
4. Para ver el texto de la anotación active el icono **Mostrar anotaciones** en la barra de herramientas Opciones de vista.

Nota: puede ver el texto de la notación aunque el icono **Mostrar anotaciones** no esté activo: pase el cursor por encima de la conexión y el texto de la anotación aparecerá en un mensaje emergente, pero solamente si está activo el icono **Mostrar información rápida** de la barra de herramientas Opciones de vista.

4.3.4 Configuración de las conexiones

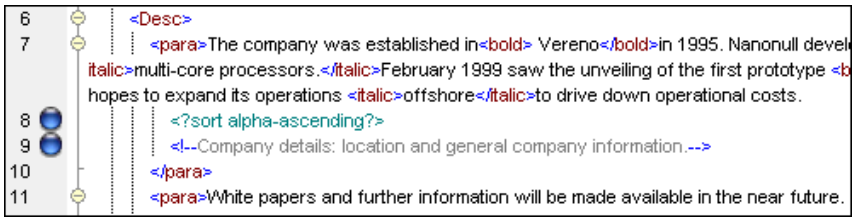
El cuadro de diálogo "Configuración de la conexión" se abre al hacer doble clic en una conexión o al hacer clic con el botón derecho en la conexión y elegir el comando **Propiedades** en el menú contextual. En este cuadro de diálogo puede definir opciones de configuración de la conexión (excepto aquellas que no estén disponibles, que aparecerán deshabilitadas).



Cuadro de diálogo "Configuración de la conexión"

En el caso de elementos de tipo `complexType` podrá elegir entre los tipos de conexión que aparecen en esta tabla (es el caso también de los elementos de tipo `complexType` que no tienen nodos de texto):

Basada en destino (estándar)	Cambia la conexión a un tipo de conexión basada en destino (*véase* *Conexiones basadas en el destino*).
Copia total (copia los elementos secundarios)	Cambia la conexión a un tipo de conexión de copia total y conecta automáticamente todos los elementos que sean idénticos en el componente de origen y destino (*véase* *Conexiones de copia total*).
Basada en origen (contenido mixto)	Cambia la conexión a un tipo de conexión basada en origen y habilita la selección de más elementos. Estos elementos adicionales deben ser elementos secundarios del elemento del archivo XML de origen que se está conectando o de lo contrario no se podrán conectar. Marque las casillas *Asignar instrucciones de procesamiento* y *Asignar comentarios* para incluir estos grupos de datos en el archivo de salida.

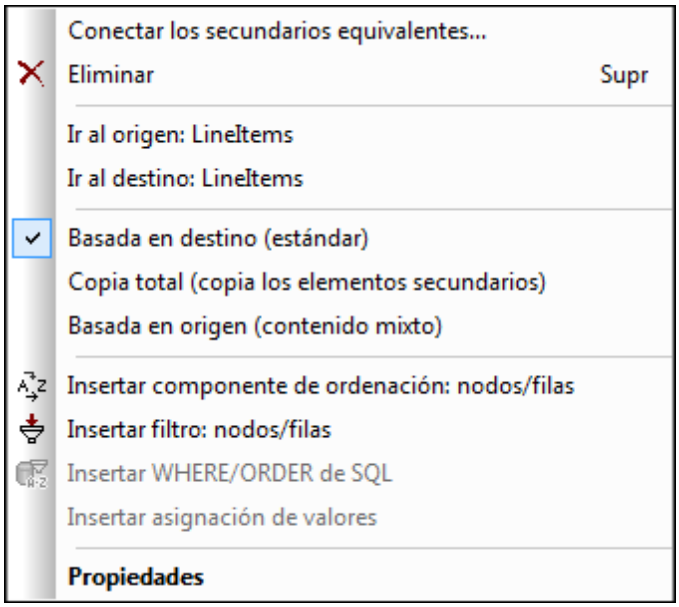


Nota: las secciones CDATA se tratan como si fueran texto.

En el grupo de opciones *Configuración de la anotación* puede especificar cómo se debe anotar la conexión (*véase* *Crear anotaciones en las conexiones*).

4.3.5 Menú contextual de las conexiones

Cuando se hace clic con el botón derecho en una conexión aparece este menú contextual:



Conectar lo secundarios equivalentes...	Abre el cuadro de diálogo "Conectar los secundarios equivalentes" (*véase* *Conectar los secundarios equivalentes*). Este comando se habilita si la conexión puede tener secundarios equivalentes.
Eliminar	Elimina la conexión seleccionada.
Ir al origen <nombre del elemento>	Selecciona el conector de origen de la conexión seleccionada.

Ir al destino <nombre del elemento>	Selecciona el conector de destino de la conexión seleccionada.
Basada en destino (estándar)	Cambia la conexión a un tipo de conexión basada en destino (*véase* *Conexiones basadas en el destino*).
Copia total (copia lo elementos secundarios)	Cambia la conexión a un tipo de conexión de copia total y conecta automáticamente todos los elementos que sean idénticos en el componente de origen y destino (*véase* *Conexiones de copia total*). Este comando solamente se habilita si el elemento de origen y el elemento de destino tienen secundarios.
Basada en origen (contenido mixto)	Cambia la conexión a un tipo de conexión basada en origen (*véase* *Conexiones basadas en el origen*). Este comando solamente se habilita si el elemento de origen y el elemento de destino tienen secundarios.
Insertar componente de ordenación: nodos/filas	Añade un componente de ordenación entre el elemento de origen y el elemento de destino (*véase* *Ordenar datos*).
Insertar filtro: nodos/filas	Añade un componente de filtrado entre el elemento de origen y el elemento de destino (*véase* *Filtros y condiciones*).
Insertar asignación de valores	Añade un componente de asignación de valores entre el elemento de origen y el elemento de destino (*véase* *Usar asignaciones de valores*).
Propiedades	Abre el cuadro de diálogo "Configuración de la conexión" (*Configuración de las conexiones*).

4.3.6 Conectar los secundarios equivalentes

Puede crear varias conexiones entre elementos que tengan el mismo nombre en el componente de origen y de destino. No obstante, debe tener en cuenta que la aplicación crea por defecto una conexión de copia total (*véase* *Conexiones de copia total*).

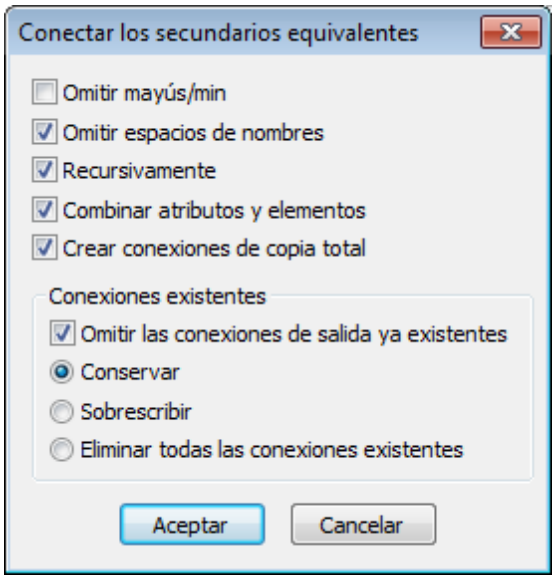
Para activar/desactivar la opción Conectar automáticamente los secundarios equivalentes:

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

Para configurar la opción Conectar los secundarios equivalentes:

1. Conecte dos elementos (de nivel superior) que tengan **elementos secundarios** con nombres idénticos en el componente de origen y de destino.
2. Haga clic con el botón derecho en la línea de conexión y elija el comando **Conectar los**

secundarios equivalentes del menú contextual.



3. Elija la configuración correspondiente (*ver tabla más abajo*) y haga clic en **Aceptar**. La aplicación crea conexiones para todos los elementos secundarios que tengan nombres idénticos utilizando la configuración definida en el cuadro de diálogo anterior.

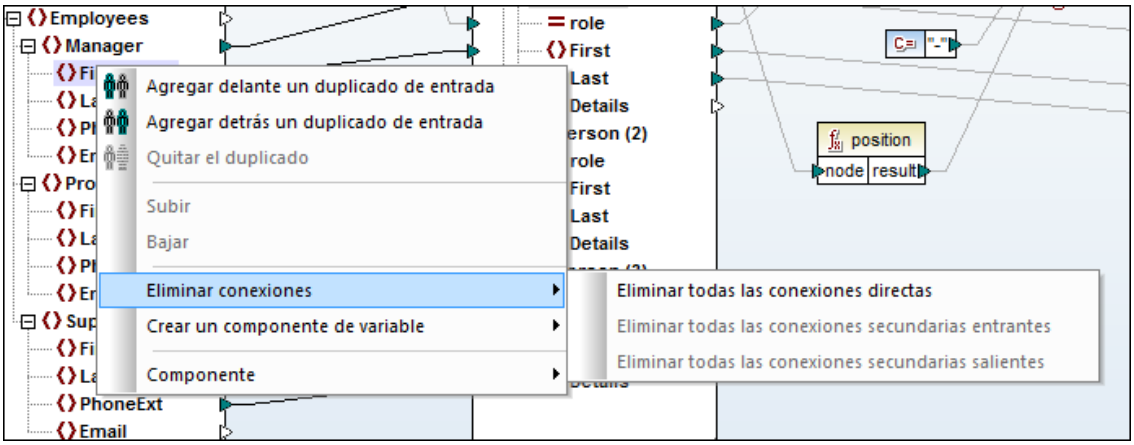
Nota: la configuración que defina en este cuadro de diálogo se utiliza al crear una conexión entre dos elementos si está activado el botón **Conectar automáticamente los secundarios equivalentes** de la barra de herramientas.

Omitir mayús/min	No tiene en cuenta el uso de mayúsculas y minúsculas a la hora de conectar secundarios equivalentes.
Omitir espacios de nombres	No tiene en cuenta los espacios de nombres a la hora de conectar secundarios equivalentes.
Recursivamente	Crea conexiones nuevas entre los secundarios equivalentes de forma recursiva. Es decir, se crea una conexión independientemente del nivel en el que los elementos estén anidados en la jerarquía, siempre y cuando tengan nombres idénticos.
Combinar atributos y elementos	Si marca esta casilla, se crearán conexiones entre atributos y elementos que tengan nombres idénticos. Por ejemplo, se puede crear una conexión si existen dos ítems llamados `Name`, aunque uno sea un elemento y el otro un atributo.
Crear conexiones de copia total	Esta casilla está activada por defecto. La aplicación crea una conexión de copia total entre los elementos de origen y destino, siempre que sea posible.
Omitir las conexiones de salida ya existentes	Crea más conexiones para todos los secundarios equivalentes, aunque ya tengan conexiones de salida.

Conservar	Conserva las conexiones ya existentes.
Sobrescribir	Vuelve a crear conexiones basadas en la configuración definida. Las conexiones ya existentes se descartan.
Eliminar todas las conexiones existentes	Elimina todas las conexiones existentes antes de crear conexiones nuevas.

Eliminar conexiones

Las conexiones creadas con ayuda del cuadro de diálogo "Conectar los secundarios equivalentes" o durante el proceso de asignación pueden eliminarse en grupo.



Para eliminar conexiones:

1. Haga clic con el botón derecho en el nombre de un elemento (no en la conexión) dentro del componente.
2. Seleccione el comando **Eliminar conexiones | Eliminar todas las conexiones...** que corresponda.

Eliminar todas las conexiones directas	Elimina todas las conexiones directas salientes o entrantes del componente actual con los demás componentes de origen o destino.
Eliminar todas las conexiones secundarias entrantes	Este comando solamente está disponible si se ejecuta sobre un elemento de un componente de destino. Elimina todas las conexiones secundarias entrantes.
Eliminar todas las conexiones secundarias salientes	Este comando solamente está disponible si se ejecuta sobre un elemento de un componente de origen. Elimina todas las conexiones secundarias salientes.

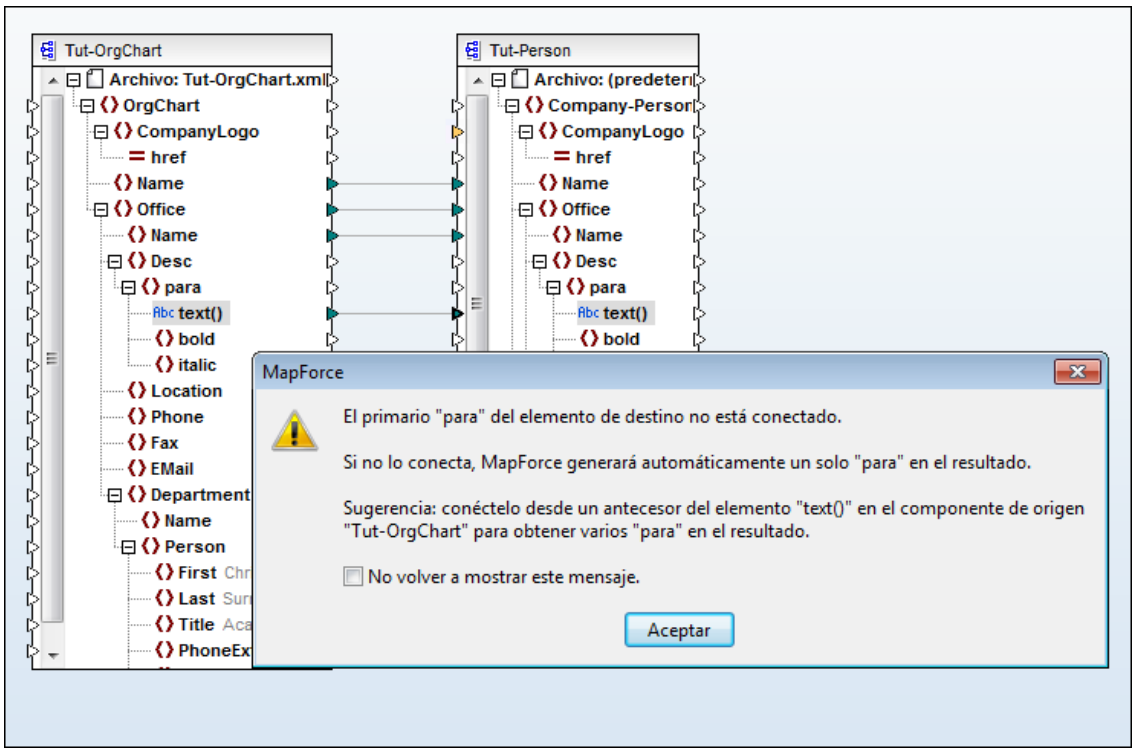
4.3.7 Notificaciones sobre conexiones antecesoras ausentes

Cuando se crean conexiones entre elementos de origen y destino de forma manual, MapForce analiza automáticamente los posibles resultados de la asignación. Si se crea una asignación entre dos elementos secundarios, MapForce puede emitir una notificación para sugerir una

conexión entre el elemento primario del elemento de origen y el elemento primario del elemento de destino.

Esta notificación le ayudará a evitar que en la ventana de vista previa de resultados aparezca un solo elemento secundario. Esto es lo que suele ocurrir si el nodo de origen ofrece una secuencia en lugar de un valor.

Veamos un ejemplo en la asignación de muestra llamada **Tut-OrgChart.mfd** (disponible en la carpeta `<Documentos>\Altova\MapForce2018\MapForceExamples\Tutorial\`). Si conecta el elemento de origen `text()` con el elemento de destino `text()`, aparece una notificación advirtiendo que el elemento primario `para` no está conectado y que solamente se generará una vez en el resultado de la asignación.



Tut-OrgChart.mfd (MapForce Basic Edition)

Para generar varios elementos `para` en el resultado debemos conectar los elementos `para` de origen y destino.

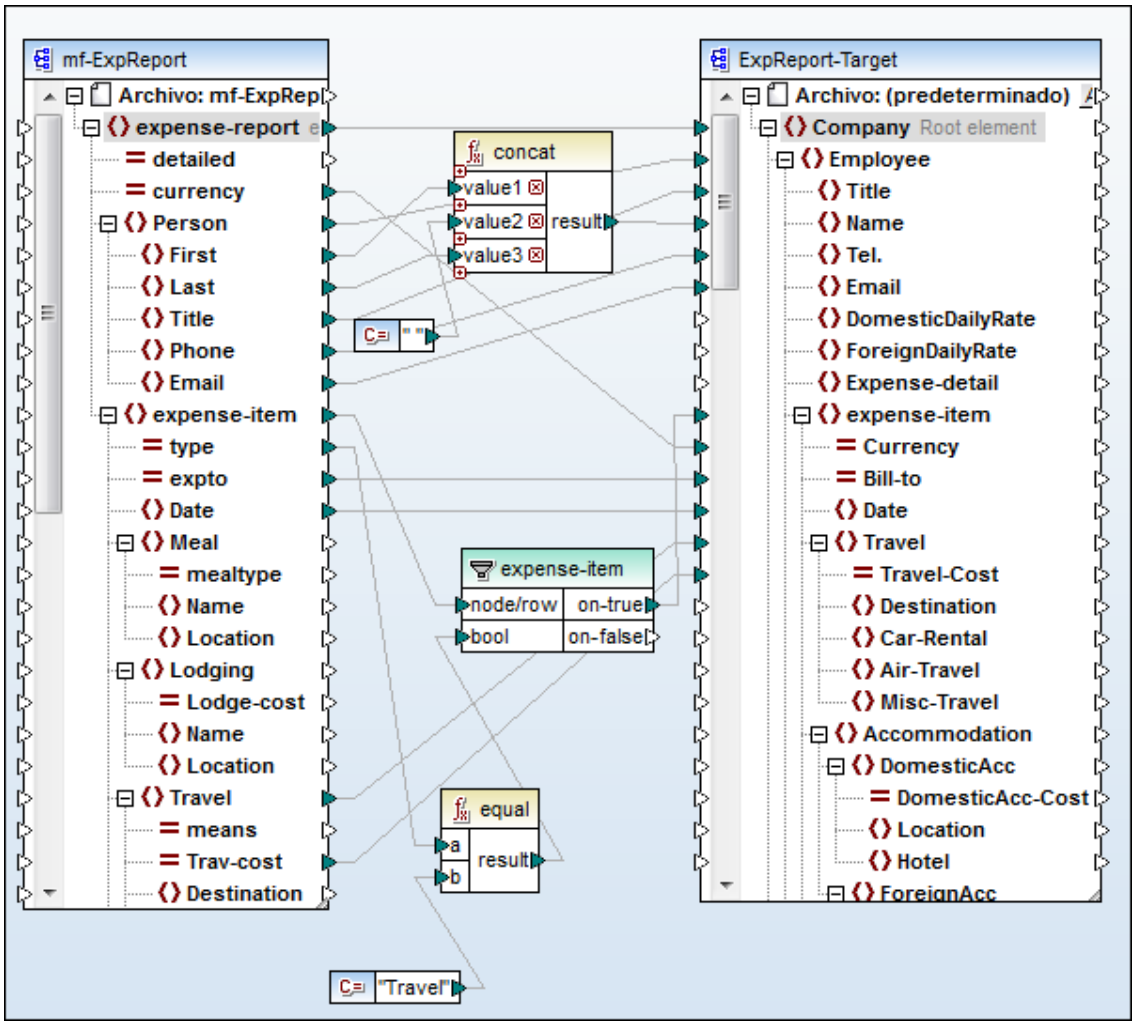
Si prefiere deshabilitar este tipo de notificaciones:

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

4.3.8 Mover conexiones y conexiones secundarias

Cuando el usuario mueve una conexión a otro componente, MapForce encuentra conexiones secundarias idénticas y pregunta si también deben moverse a la nueva posición. Esta característica puede ser útil si ya tiene un diseño de asignación y cambia el elemento raíz del esquema de destino. Cuando esto ocurre, lo normal es que sea necesario reasignar todas las conexiones descendientes a mano. Gracias a esta característica podrá evitar este tipo de situaciones.

Veamos un ejemplo en la asignación de muestra **Tut-ExpReport.mfd** (disponible en la carpeta `<Documentos>\Altova\MapForce2018\MapForceExamples\Tutorial\`).



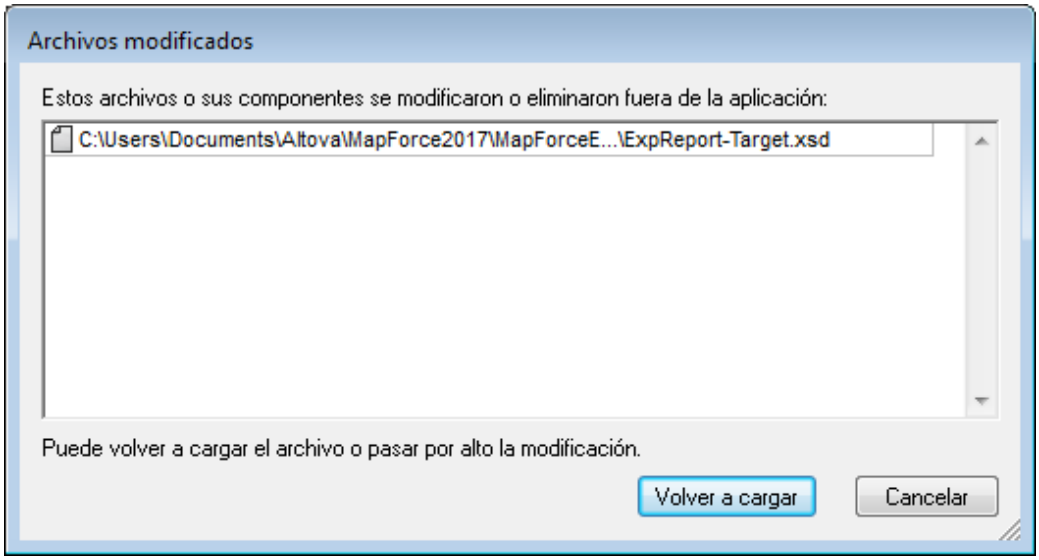
Tut-ExpReport.mfd (MapForce Basic Edition)

Hagamos una prueba para comprender el funcionamiento de esta característica:

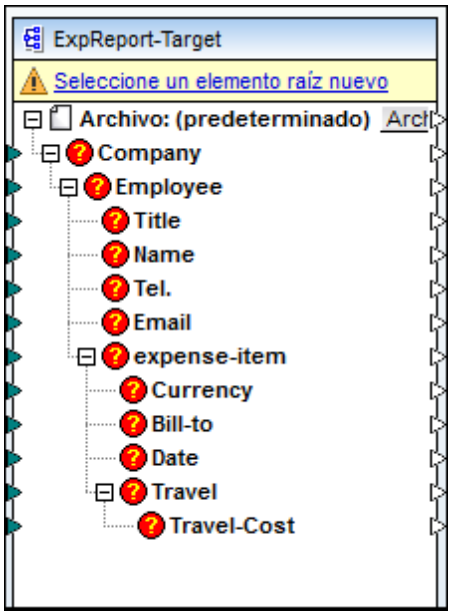
1. Abra la asignación de muestra **Tut-ExpReport.mfd**.
2. Edite el esquema **ExpReport-Target.xsd** fuera de MapForce, cambiando el elemento

raíz `Company` del esquema de destino por `Company-EU`. No es necesario cerrar MapForce para realizar este cambio.

3. Tras cambiar el elemento raíz `Company` del esquema de destino por `Company-EU`, MapForce emite esta notificación:

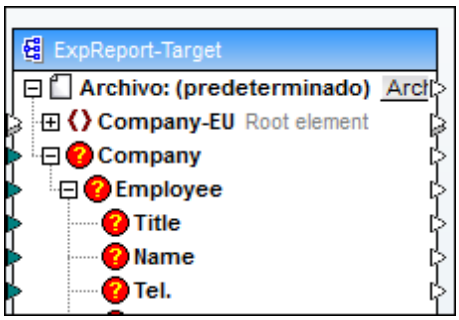


4. Haga clic en el botón **Volver a cargar** para volver a cargar el esquema que modificó. Como el elemento raíz se eliminó, el componente muestra varios nodos ausentes.

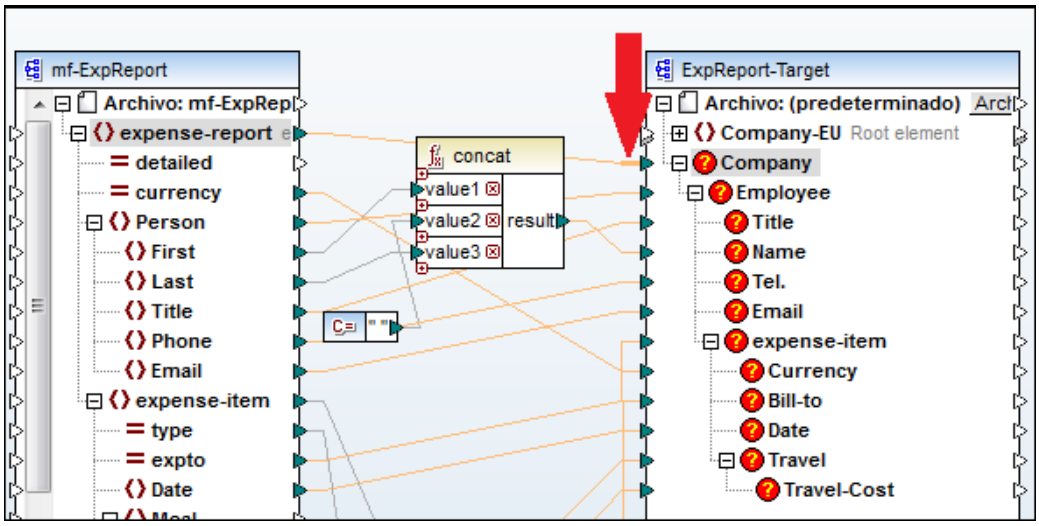


5. Haga clic en **`Seleccione un elemento raíz nuevo`** en la parte superior del componente. (También puede cambiar de elemento raíz haciendo clic con el botón derecho en el título del componente y seleccionando **Cambiar de elemento raíz** en el menú contextual.)

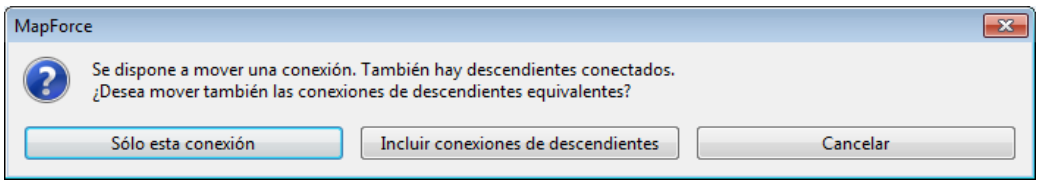
6. Seleccione `Company-EU` como nuevo elemento raíz y haga clic en **Aceptar** para confirmar. Ahora el elemento raíz `Company-EU` aparece al principio del componente.



7. Haga clic en el extremo de destino de la conexión que une el elemento `expense-report` del componente de origen y el elemento `Company` del elemento de destino. Después arrástrelo hasta el elemento raíz `Company-EU` del componente de destino.



Ahora aparece una notificación.



8. Haga clic en **Incluir conexiones de descendientes**. Esto da la orden para que

MapForce reasigne los elementos secundarios correctos bajo el nuevo elemento raíz y la asignación vuelva a ser válida.

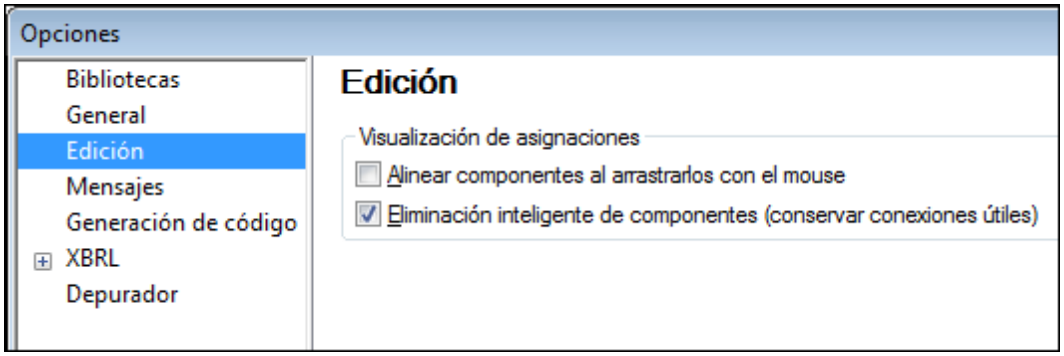
Nota: si el nodo con el que crea la asignación tiene el mismo nombre que el nodo de origen pero un espacio de nombres distinto, la notificación incluirá un botón más llamado **Incluir descendientes y asignar espacio de nombres**. Haga clic en este botón para mover las conexiones secundarias que estén en el mismo espacio de nombres que el nodo primario de origen a los mismos nodos secundarios situados bajo el nodo que tiene diferente espacio de nombres.

4.3.9 Conservar conexiones tras eliminación de componentes

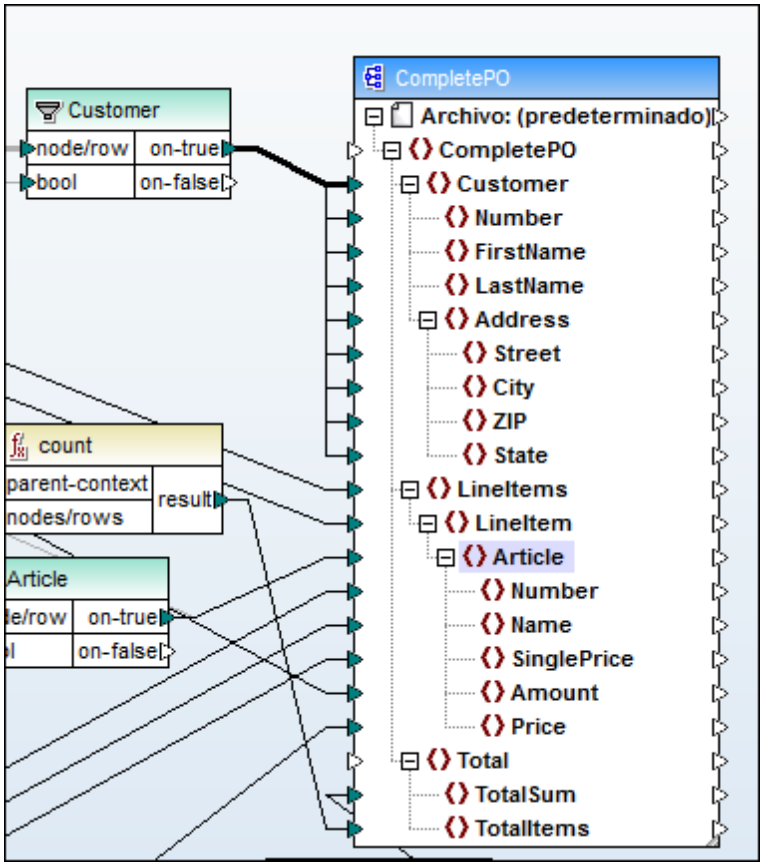
Tiene la opción de elegir qué ocurre cuando se elimina un componente que tiene varias conexiones (secundarias) con otro componente, como cuando elimina un filtro o un componente de ordenación, por ejemplo. Esto puede ser de gran utilidad si quiere conservar todas las conexiones secundarias para no tener que restaurarlas una a una.

Tras eliminar el componente tendrá la opción de conservar o restaurar las conexiones secundarias o de eliminar todas inmediatamente.

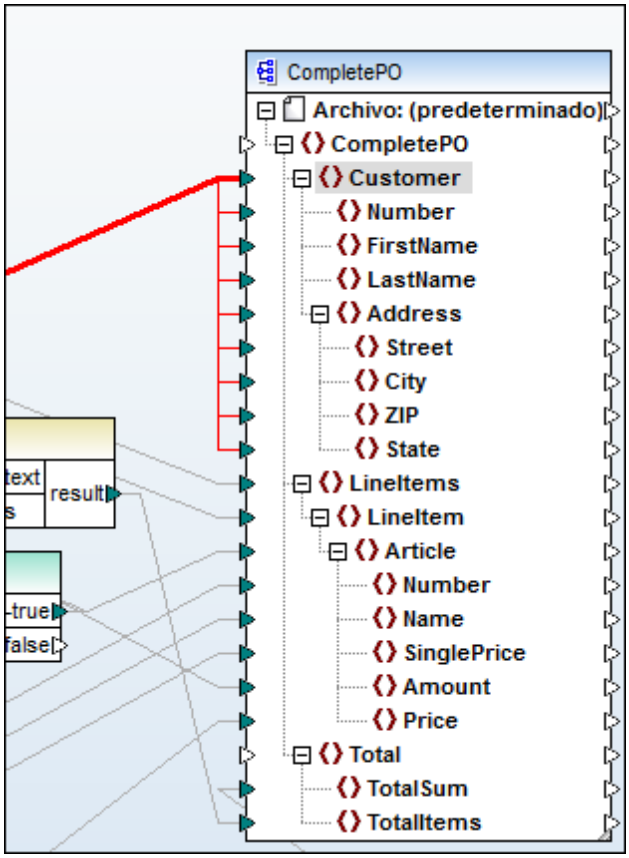
Seleccione la pestaña **Edición** del cuadro de diálogo "Opciones" para ver la configuración actual. En la configuración predeterminada la casilla *Eliminación inteligente de componentes (conservar conexiones útiles)* está desactivada.



Veamos un ejemplo en la asignación de muestra **CompletePO.mfd** (disponible en la carpeta `...\MapForceExamples`) con la casilla *Eliminación inteligente de componentes (conservar conexiones útiles)* **activada**. Como puede ver en la imagen siguiente, el filtro **`Customer`** es una conexión de **copia total** que tiene varios elementos secundarios conectados.



Si eliminamos el filtro **`Customer`** aparece una notificación preguntando si desea eliminar este componente. Si hace clic en **Sí**, el filtro se elimina pero las conexiones secundarias se conservan.



Observe que los conectores que se conservan todavía están seleccionados (es decir, están marcados en color rojo). Si desea eliminarlos, basta con pulsar la tecla **Supr**.

Haga clic en cualquier parte de la asignación para dejar de seleccionar los conectores.

Si la casilla *Eliminación inteligente de componentes (conservar conexiones útiles)* estuviera **desactivada**, al eliminar el filtro se eliminarían también todas sus conexiones secundarias.

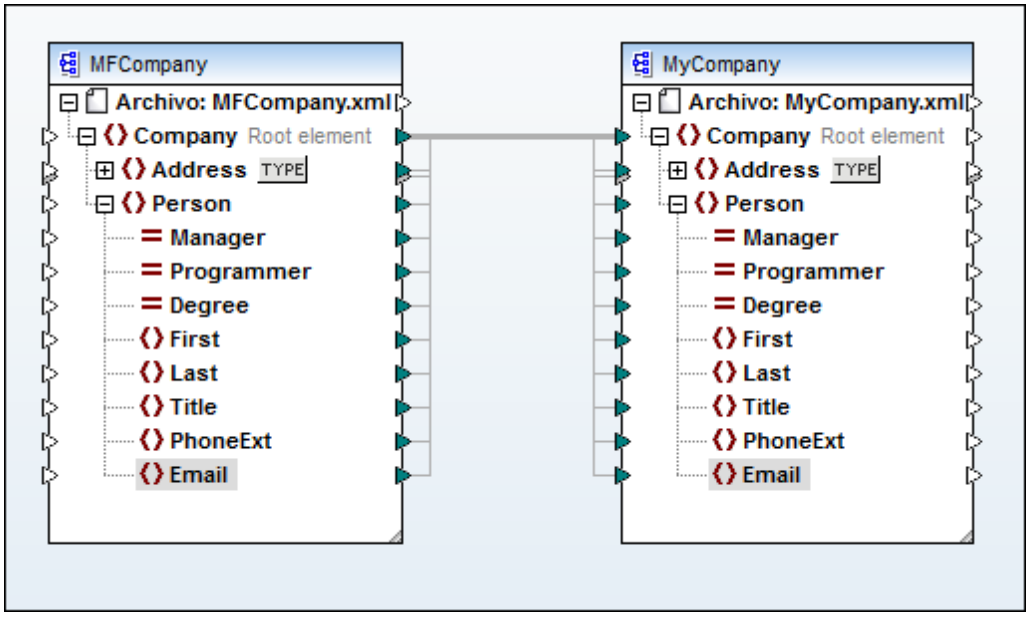
Nota: si un componente de filtrado tiene conectadas las dos salidas (**`on-true`** y **`on-false`**), entonces se conservan las conexiones secundarias de ambas salidas.

4.3.10 Nota sobre elementos ausentes

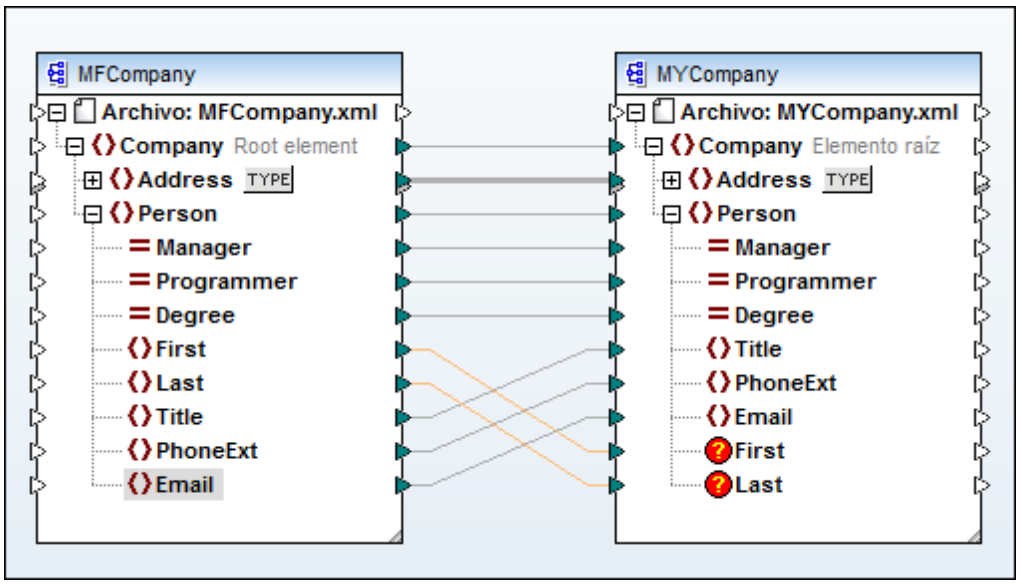
A medida que pasa el tiempo lo más probable es que la estructura de uno de los componentes de la asignación de datos sufra cambios (p. ej. es posible que se añadan o eliminen elementos o atributos en un esquema XML). MapForce utiliza elementos marcadores de posición para conservar todos los conectores y todos los datos de conexión entre componentes.

Ejemplo

Para este ejemplo usaremos el archivo de esquema **MFCompany.xsd**. Cambiemos el nombre del esquema por `MyCompany.xsd` y creemos un conector entre el elemento `Company` disponible en ambos esquemas. Esta operación crea conectores para todos los elementos secundarios de los componentes (si está activada la opción **Conectar los secundarios equivalentes**).



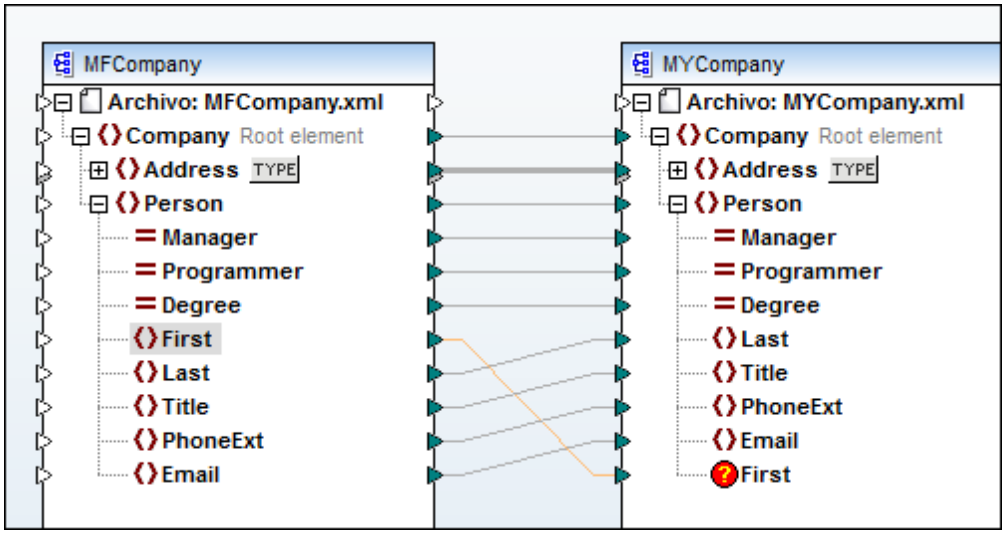
Mientras editamos `MyCompany.xsd` en XMLSpy, eliminamos los elementos `First` y `Last` del esquema. Al volver a MapForce aparece una notificación solicitando que volvamos a cargar el esquema. Al hacer clic en **Volver a cargar** se actualizan los componentes en MapForce.



Ahora los elementos eliminados y sus conectores aparecen marcados en el componente **MyCompany**. Ahora tenemos la opción de volver a conectar los conectores con otros elementos o eliminarlos.

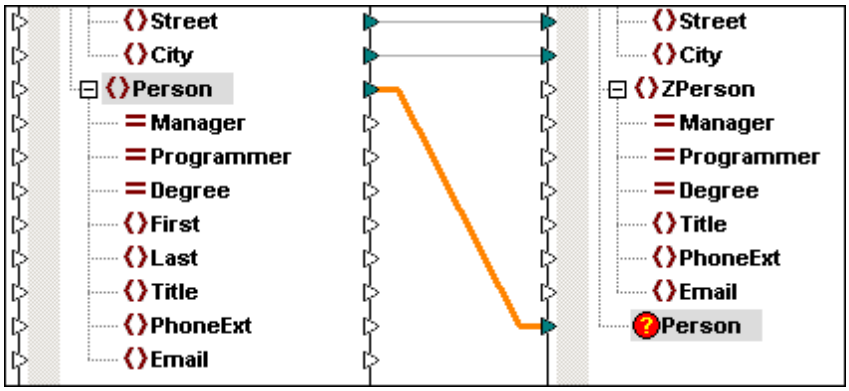
Recuerde que podrá obtener una vista previa de la asignación (y generar código) pero la ventana Mensajes emitirá advertencias si lo hace en este momento. Todas las conexiones entrantes y salientes de elementos ausentes se omiten durante la generación de código y de vista previa.

si hace clic en un conector resaltado y lo elimina, el elemento ausente se elimina del componente (p. ej. `Last` en el componente **MyCompany**).



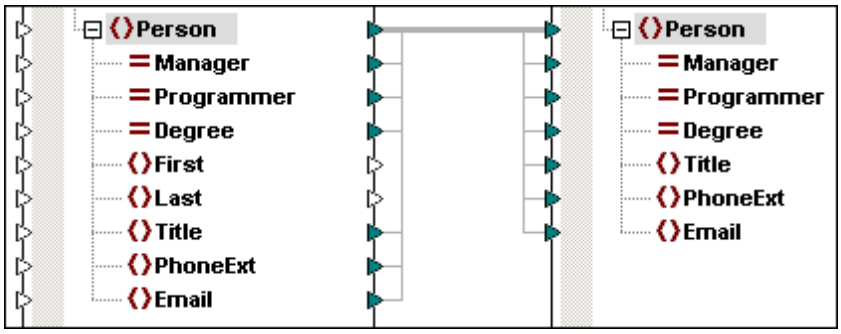
Cambiar el nombre de los elementos

Si cambia el nombre de un elemento primario (p. ej. cambiamos `Person` por `ZPerson`), entonces se conserva el conector del antecesor original pero sus elementos secundarios y conectores se eliminan.



Conectores de copia total y elementos ausentes

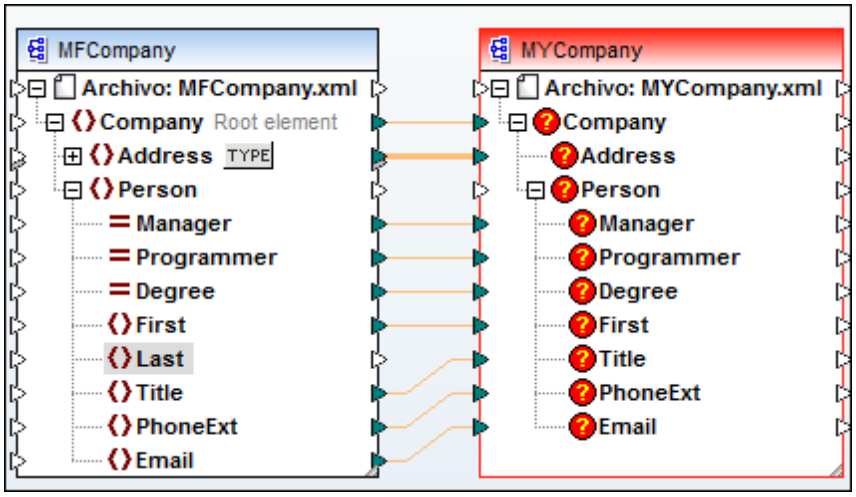
Las conexiones de copia total se tratan como si fueran conexiones normales. La única diferencia es que lo conectores con elementos secundarios ausentes no se conservan ni señalan.



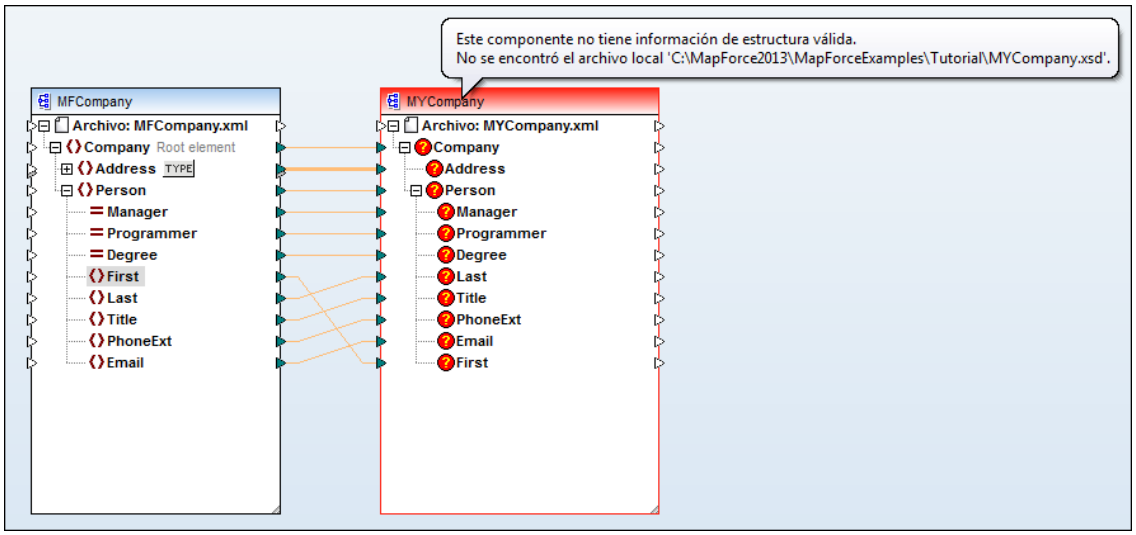
Orígenes de datos renombrados o eliminados

Si el **origen de datos** de un componente (es decir, el esquema) cambió de nombre o se eliminó, entonces la aplicación resalta todos los elementos que contenía. Cuando la aplicación marque el

recuadro de un componente en color rojo, significa que falta una conexión válida a un archivo de esquema, lo cual impide la generación de la vista previa y de código.



Al pasar el cursor por encima del componente resaltado aparece un mensaje de información rápida.



Al hacer doble clic en la barra de título del componente resaltado aparece el cuadro de diálogo "Configuración del componente". Haga clic en el botón **Examinar** del grupo de opciones *Archivo de esquema* para seleccionar un esquema distinto o una copia de seguridad del esquema original. Para más información consulte la sección dedicada al menú de comandos **Componente**.

La aplicación conservará todas las conexiones válidas/correctas si selecciona un esquema con la misma estructura.

Altova MapForce 2018 Basic Edition

Diseño de asignaciones

5 Diseño de asignaciones

Sitio web de Altova: Herramienta de integración de datos

En esta sección encontrará información para trabajar con diseños de asignación de datos e instrucciones para transformar datos en el área de asignación.

Si su objetivo es....	Consulte este apartado
Crear o editar referencias de rutas de acceso a archivos de esquema, de instancia, etc. utilizados por una asignación de datos.	Usar rutas de acceso absolutas y relativas
Adaptar la asignación de datos a requisitos específicos (p. ej. para ajustar la secuencia de los nodos en un componente de destino).	Tipos de conexión
Usar el resultado de un componente como entrada de otro componente.	Asignaciones en cadena y componentes de paso a través
Procesar varios archivos (p. ej. todos los archivos de un directorio) en una misma asignación, como origen o como destino.	Procesar varios archivos de entrada o salida simultáneamente
Pasar un valor externo (p. ej. un parámetro de cadena) a la asignación.	Pasar parámetros a la asignación
Obtener como resultado de la asignación un valor de cadena en lugar de un archivo.	Obtener valores de cadena de una asignación
Almacenar ciertos datos de asignación temporalmente para procesarlos más adelante (similar al concepto de variable en los lenguajes de programación).	Usar variables
Ordenar datos de forma ascendente o descendente.	Ordenar datos
Filtrar nodos/filas en función de determinados criterios o procesar valores de forma condicional.	Filtros y condiciones
Combinar datos de varios componentes de origen con esquemas distintos.	Combinar datos de varios esquemas distintos
Procesar pares de clave/valor. Por ejemplo, para convertir los meses de una representación numérica (01, 02, etc.) en una representación textual (enero, febrero, etc.).	Usar asignaciones de valores
Aprender a evitar resultados no deseados en asignaciones de datos complejas.	Reglas y estrategias de asignación de datos

Nota importante: MapForce viene con una amplia biblioteca de funciones integradas (*véase Referencia de la biblioteca de funciones*) diseñada para ayudar al usuario con una multitud de tareas de procesamiento. Si esta biblioteca de funciones integradas no es suficiente, puede crear sus propias funciones en MapForce o reciclar archivos XSLT externos. Para más información consulte la sección Funciones.

5.1 Usar rutas de acceso absolutas y relativas

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 para validarlos. Por su parte, los archivos de instancia sirven para leer los datos de origen, validarlos con el esquema y generar una vista previa de la asignación de datos.

Cuando añadimos un componente a la asignación, MapForce crea las 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. También describimos las diferencias que existen entre usar rutas de acceso relativas y absolutas.

5.1.1 Usar rutas de acceso relativas en un componente

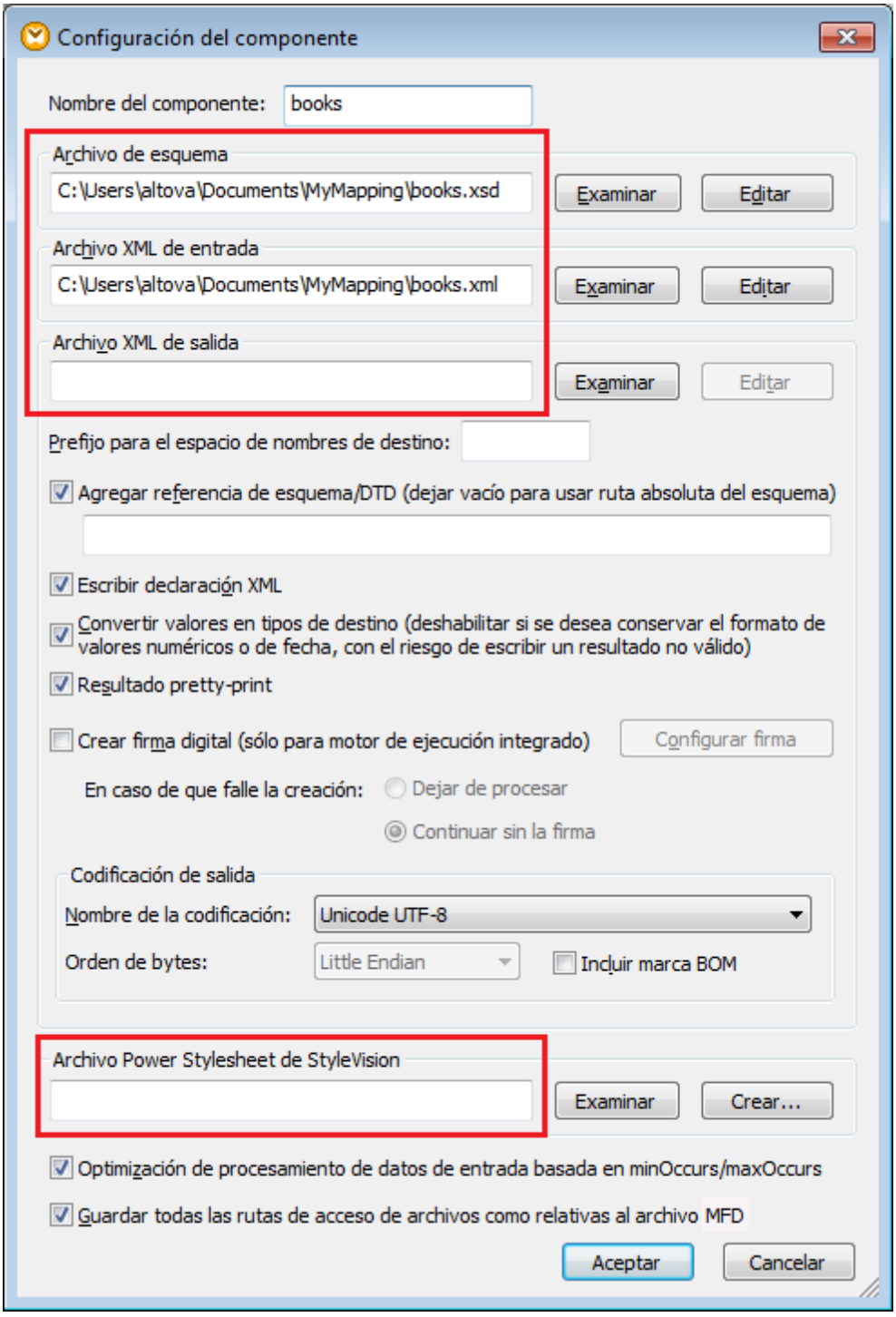
En el cuadro de diálogo "Configuración del componente" podemos indicar rutas de acceso relativas o absolutas para los diferentes archivos a los que puede hacer referencia el componente:

- Archivos de entrada (es decir, archivos en los que MapForce lee datos).
- Archivos de salida (es decir, archivos en los que MapForce escribe datos).
- Archivos de esquema (si el componente tiene un esquema).
- Archivos de estructura (si el componente tiene estructuras complejas, como parámetros de entrada o salida de funciones definidas por el usuario o variables).
- Archivos StyleVision Power Stylesheet (*.sps) utilizados para aplicar formato a los datos y generar documentos de salida en PDF, HTML y Word.

Puede introducir rutas de acceso relativas en los diferentes cuadros de texto directamente (marcados en rojo en la imagen siguiente).

> Antes de introducir las rutas de acceso relativas, asegúrese de guardar el archivo de asignación de datos (.mfd). Si no lo hace, todas las rutas de acceso relativas se resolverán a partir de la carpeta de aplicación personal de Windows (`Documentos\Altova\MapForce2018`).

También puede hacer que MapForce guarde todas las rutas de acceso mencionadas anteriormente como relativas al archivo de asignación (.mfd). Por ejemplo, en la imagen siguiente puede ver que al final del cuadro de diálogo "Configuración del componente" hay una casilla llamada *Guardar todas las rutas de acceso de archivos como relativas al archivo MFD.* Si esta casilla está marcada (opción predeterminada y recomendada), las rutas de acceso de los archivos a los que hace referencia el componente se guardarán como relativas a la ruta de acceso del archivo de diseño de asignación (.mfd). Esto afecta a todos los archivos a los que hace referencia el componente (marcados con un recuadro rojo en la imagen siguiente).



Cuadro de diálogo "Configuración del componente"

En el ejemplo anterior se trataba de un componente XML, pero la casilla *Guardar todas las rutas de acceso de archivos como relativas al archivo MFD* tiene el mismo funcionamiento con estos archivos:

- Archivos de estructuras utilizados por parámetros de entrada o salida complejos de funciones definidas por el usuario o por variables de tipo complejo.
- Archivos planos de entrada o salida*.
- Archivos de esquema a los que hacen referencia componentes de BD compatibles con campos XML*.
- Archivos de entrada o salida XBRL, FlexText, EDI, Excel 2007+ y JSON files.

* MapForce Professional y Enterprise Edition
** MapForce Enterprise Edition solamente

Volvamos al componente XML del ejemplo. Si el archivo .mfd está en la misma carpeta que los archivos **books.xsd** y **books.xml**, MapForce realizará estos cambios eb las rutas de acceso:

C:\Usuarios\altova\Documentos\MyMapping\books.xsd se cambiará por **books.xsd**
C:\Usuarios\altova\Documentos\MyMapping\books.xml se cambiará por **books.xml**

Las rutas de acceso que hagan referencia a unidades de disco no locales o que utilicen una URL no se convertirán en rutas relativas.

Cuando la casilla *Guardar todas las rutas de acceso de archivos como relativas al archivo MFD* esté marcada, MapForce también vigilará los archivos a los que hace referencia el componente cuando la asignación se guarde en una carpeta nueva con el comando de menú **Guardar como**. Además, si todos los archivos están en la misma carpeta que la asignación, las referencias de rutas de acceso no se romperán cuando mueva el directorio entero a una ubicación nueva en el disco.

El uso de rutas de acceso relativas (y, por tanto, la activación de la casilla *Guardar todas las rutas de acceso de archivos como relativas al archivo MFD*) puede se muy importante en algunos casos. Por ejemplo:

- Si es probable que cambie la ubicación de la asignación en el sistema operativo.
- Si la asignación está en un directorio que está bajo control de código fuente (controlado por un sistema de control de código fuente como TortoiseSVN, por ejemplo).
- Si tiene pensado implementar la asignación en un equipo distinto (e incluso en un sistema operativo distinto) para ejecutarla.

Si no está marcada la casilla *Guardar todas las rutas de acceso de archivos como relativas al archivo MFD*, cuando guarde la asignación no se modificarán las rutas de acceso de los archivos (es decir, quedarán tal y como aparecen en el cuadro de diálogo "Configuración del componente").

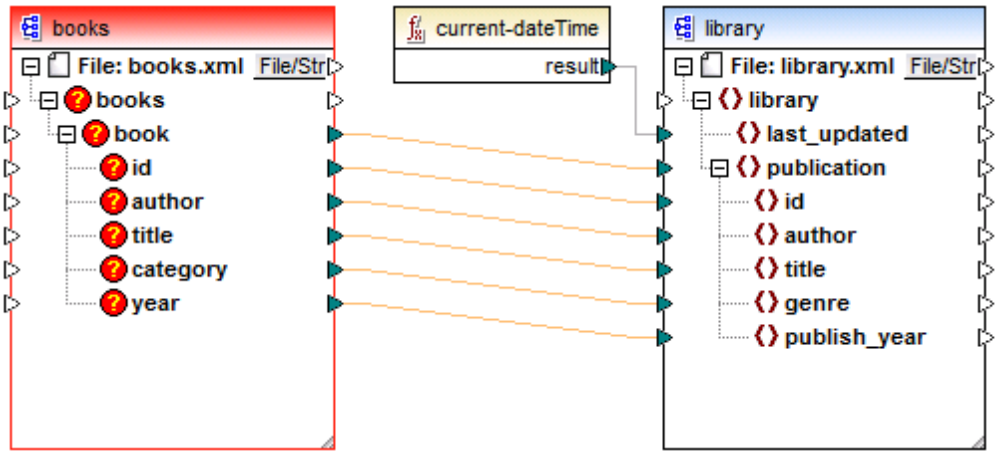
5.1.2 Corregir referencias a rutas de acceso rotas

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. De este modo MapForce reduce el riesgo de que haya referencias de ruta de acceso rotas. 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. Por ejemplo:



Referencia de ruta de acceso rota

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" (*véase* *Cambiar configuración de los componentes*).

5.1.3 Rutas de acceso según el entorno de ejecución

Si genera código a partir de la asignación de datos, los archivos generados ya no se ejecutan en MapForce, sino en el entorno de destino que usted elija (p. ej. RaptorXML Server). Sin embargo, para que la asignación de datos pueda ejecutarse correctamente es imprescindible que las rutas de acceso relativas sean significativas en el entorno donde se ejecuta la asignación.

Por tanto, cuando la asignación utilice rutas de acceso relativas de archivos de instancia o esquema, tenga en cuenta estas rutas de acceso base para cada lenguaje de destino:

Lenguaje de destino	Ruta de acceso base
XSLT/XSLT2	Ruta de acceso del archivo XSLT.
XQuery*	Ruta de acceso del archivo XQuery.
C++, C#, Java*	Directorio de trabajo de la aplicación generada.
BUILT-IN* (cuando genere vista previa de la asignación en MapForce)	Ruta de acceso del archivo de asignación (.mfd).
BUILT-IN* (cuando ejecute la asignación con MapForce Server)	El directorio de trabajo actual.

Lenguaje de destino	Ruta de acceso base
BUILT-IN* (cuando ejecute la asignación con MapForce Server bajo control de FlowForce Server)	El directorio de trabajo del trabajo o de FlowForce Server.

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

Si lo necesita, también puede hacer que MapForce convierta todas las rutas de acceso relativas en rutas de acceso absolutas a la hora de generar código para una asignación. Esta opción puede ser muy útil si el código de asignación se ejecuta en el mismo sistema operativo o incluso en otro sistema operativo donde se puedan resolver las rutas de acceso absolutas utilizadas por la asignación.

Para convertir todas las rutas de acceso en absolutas en el código generado basta con marcar la casilla *Convertir las rutas de acceso en absolutas en el código generado* en el cuadro de diálogo "Configurar asignación" (*véase* *Cambiar configuración de la asignación*).

Cuando genere código con esta casilla activada, MapForce resolverá las rutas de acceso relativas en base al directorio del archivo de asignación (.mfd) y las convertirá en absolutas en el código generado. Esta configuración afectará a la ruta de acceso de estos archivos:

- Archivos de instancia de entrada y salida para todas las clases de componente basados en archivos.

Cuando esté desactivada la casilla *Convertir las rutas de acceso en absolutas en el código generado,* las rutas de acceso se conservarán tal y como se definieran en la configuración del componente.

5.1.4 Operaciones cortar y pegar con rutas de acceso relativas

Cuando copie un componente de una asignación y lo pegue en otra, MapForce comprueba que las rutas de acceso relativas de los archivos de esquema se puedan resolver con la carpeta de la asignación de destino. Si las rutas de acceso no se pueden resolver, la aplicación solicitará que convierta las rutas de acceso relativas en absolutas a través de la carpeta de la asignación de origen. Se recomienda guardar en primer lugar la asignación de destino porque, de lo contrario, las rutas de acceso relativas se resolverán con la carpeta de la aplicación personal.

5.2 Tipos de conexión

Cuando cree una conexión de asignación (y tanto el nodo de origen como el de destino tengan secundarios) tendrá la opción de elegir entre tres tipos de conexiones:

- Conexión basada en el destino (estándar).
- Conexión basada en el origen (contenido mixto).
- Conexión de copia total (copia de secundarios).

El tipo de conexión determina la secuencia de los secundarios en el resultado que se genera a partir de la asignación. En esta sección encontrará información sobre cada tipo de conexión y cómo utilizar cada uno de ellos.

5.2.1 Conexiones basadas en el destino

Cuando una conexión está basada en el destino (conexión estándar), la secuencia de los nodos secundarios en el resultado de la asignación viene dado por la secuencia que siguen los nodos en el esquema de destino. Este tipo de conexión es adecuada para la mayoría de las asignaciones y es el tipo de conexión predeterminada de MapForce.

En las asignaciones las conexiones basadas en el destino se representan por medio de una línea sólida.
On a mapping, target-driven connections are shown with a solid line.

Es posible que las conexiones basadas en el destino no sean adecuadas cuando su intención sea asignar nodos XML que tienen contenido mixto (datos de caracteres y elementos secundarios). Por ejemplo:

```
<p>Este es nuestro producto <i>más vendido</i>.</p>
```

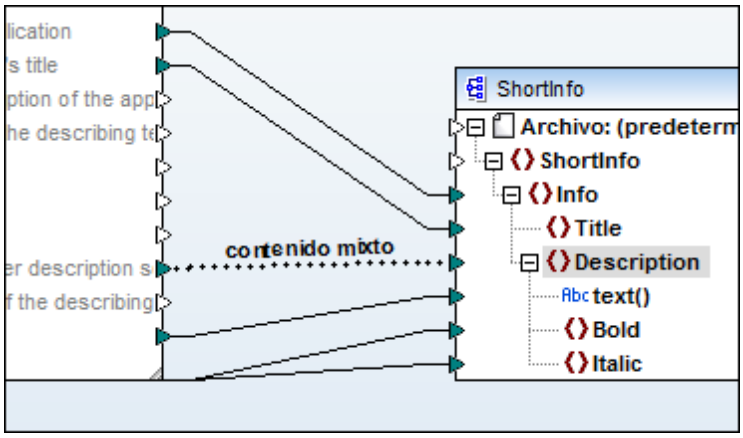
Cuando los nodos tienen contenido mixto, es probable que el objetivo sea conservar la secuencia de los nodos tal y como aparecen en el archivo de origen. Para ello se recomienda el uso de conexiones basadas en el origen (*véase* *el apartado siguiente*).

5.2.2 Conexiones basadas en el origen

Las conexiones basadas en el origen (contenido mixto) permiten asignar automáticamente nodos de texto y nodos secundarios en la misma secuencia en que aparecen en el archivo XML de **origen**.

- El contenido de nodos de texto se puede asignar.

- La secuencia de los nodos secundarios depende del archivo XML de instancia de origen.



En las asignaciones las conexiones de contenido mixto se presentan por medio de una línea de puntos.

Las conexiones basadas en el origen o de contenido mixto también se pueden aplicar a elementos **complextType**. Los nodos secundarios se asignarán en base a su secuencia en el archivo XML de origen.

Las conexiones basadas en el origen o de contenido mixto son compatibles con asignaciones:

- De componentes de **origen**:
  - elementos complexType del esquema XML (incluidos elementos de contenido mixto, es decir mixed=true)
- De componentes de **destino**:
  - elementos complexType del esquema XML (incluidos elementos de contenido mixto) **Nota:** las secciones CDATA se tratan como si fueran texto.

5.2.2.1 Crear asignaciones entre contenido mixto

Los archivos utilizados para ilustrar este ejemplo son **Tut-OrgChart.mfd**, **Tut-OrgChart.mfd.xml**, **Tut-OrgChart.mfd.xsd** y **Tut-Person.xsd** y están disponibles en la carpeta `...\MapForceExamples\Tutorial\`.

Instancia XML de origen

En la imagen que aparece a continuación puede ver parte del archivo **Tut-OrgChart.xml** utilizado para ilustrar este ejemplo. Ahora mismo nos centraremos en el elemento de contenido mixto `para`, así como 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 también se pueden asignar, tal y como se puede ver más abajo.


```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2005 sp2 U (http://www.altova.com) by Mr. Nobody (Altova GmbH) -->
<OrgChart xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="Tut-OrgChart.xsd">
  <CompanyLogo href="nanonull.gif"/>
  <Name>Organization Chart</Name>
  <Office>
    <Name>Nanonull, Inc.</Name>
    <Desc>
      <para>The company was established in<bold> Vereno</bold>in 1995. Nanonull
develops nanoelectronic technologies for<italic>multi-core processors.</italic>February 1999
saw the unveiling of the first prototype <bold>Nano-grid.</bold>The company hopes to expand
its operations <italic>offshore</italic>to drive down operational costs.
        <?sort alpha-ascending?>
        <!--Company details: location and general company information.-->
      </para>
      <para>White papers and further information will be made available in the near future.
    </Desc>
```

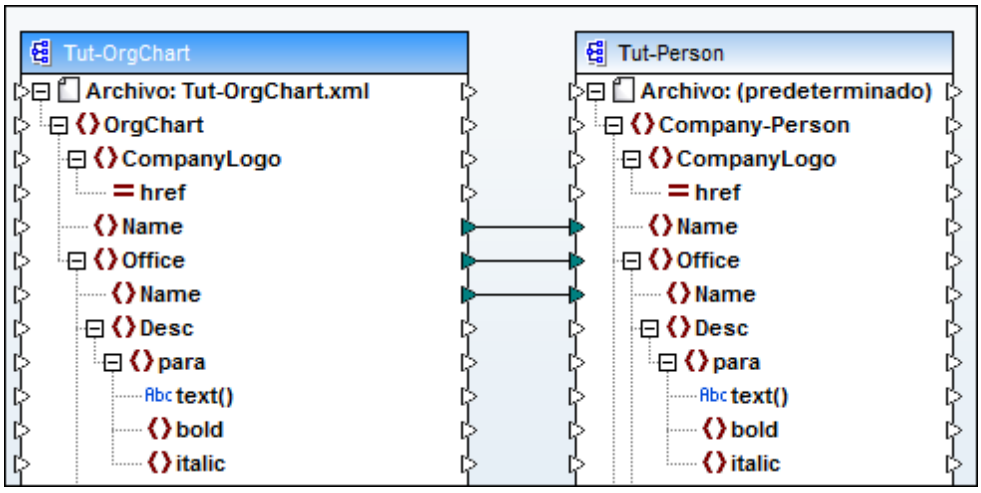
Veamos ahora qué secuencia tienen los nodos de texto y los nodos `bold` e `italic` de Nanonull., Inc en el archivo XML de instancia:

<para> The company…
<**bold**>Vereno<**/bold**>in 1995 …
<**italic**>multi-core...<**/italic**>February 1999

<**bold**>Nano-grid.<**/bold**>The company …
<**italic**>offshore...<**/italic**>to drive...
</para>

Asignación inicial

A continuación puede ver el estado inicial que tiene la asignación al abrir el archivo **Tut-Orgchart.mfd**.



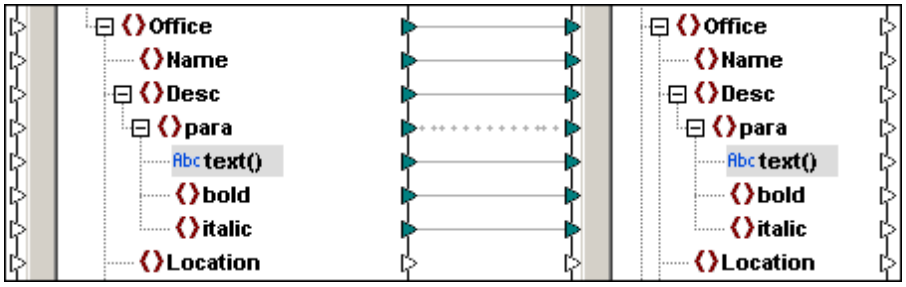
Resultado de la asignación anterior

El resultado de la asignación inicial puede verse en la imagen siguiente. En el resultado se generó `Organization Chart` junto con el nombre de las diferentes oficinas.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <Company-Person xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNames
3    <Name>Organization Chart</Name>
4    <Office>
5      <Name>Nanonull, Inc.</Name>
6    </Office>
7    <Office>
8      <Name>Nanonull Europe, AG</Name>
9    </Office>
10 </Company-Person>
11
```

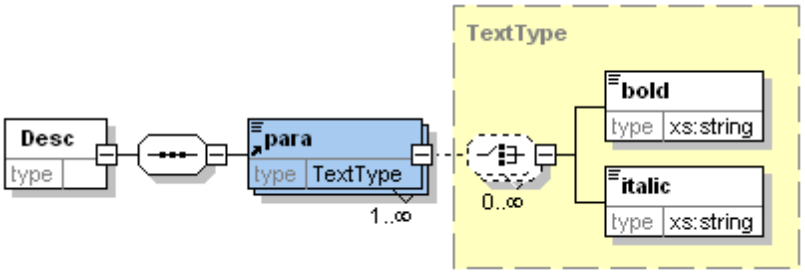
Asignación del elemento para

En la imagen siguiente puede ver un ejemplo de asignación de contenido mixto. El elemento `para` es de contenido mixto y su conexión se representa por medio de una línea de puntos. El nodo `text()` contiene los datos de texto y debe asignarse para que el texto aparezca en el componente de destino.



Para anotar una conexión (es decir, para añadirle una etiqueta) haga clic con el botón derecho en la conexión y elija el comando **Propiedades** (*véase* *Crear anotaciones en las conexiones*).

En la imagen siguiente podemos ver el modelo de contenido del elemento `Description` (`Desc`) del archivo de esquema **Tut-OrgChart.xsd**. Esta definición es idéntica en los esquemas de origen y destino utilizados en este ejemplo.



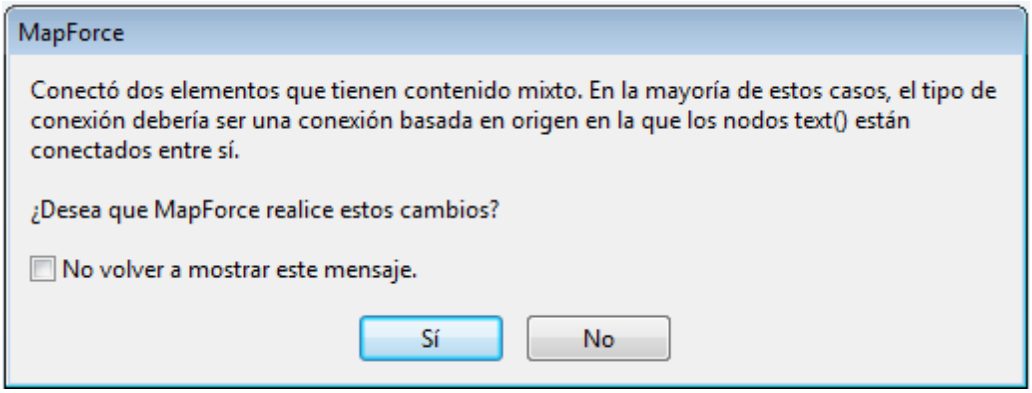
En el modelo de contenido podemos observar estas propiedades del elemento `para`:

- **para** es un elemento complexType con mixed="true", de tipo "TextType".
- los elementos **bold** e **italic** son de tipo "xs:string" y no se definieron como recursivos en este ejemplo (es decir, ni **bold** ni **italic** son de tipo "TextType").
- los elementos **bold** e **italic** pueden aparecer tantas veces como sea necesario en cualquier secuencia dentro del elemento **para**.
- dentro del elemento **para** pueden aparecer tantos nodos de texto como sean necesarios, intercalados con tantos elementos **bold** e **italic** como sean necesarios.

Para crear conexiones de contenido mixto entre elementos:

1. Seleccione el comando de menú **Conexión | Conectar automáticamente los secundarios equivalentes** para activar esta opción si aún no lo está.
2. Conecte el nodo **para** del esquema de origen con el nodo **para** del esquema de destino.

   Ahora aparece un mensaje preguntando si desea que MapForce defina los conectores como conectores basados en el origen.



3. Haga clic en **Sí** para crear una conexión de contenido mixto

   Nota: para es de contenido mixto y desencadena el mensaje en este momento dado. El mensaje de contenido mixto también aparece si solo se asignan los nodos **para** directamente, sin tener activada la opción de conexión automática de secundarios equivalentes.

   Ahora todos los elementos secundarios de **para** están conectados. El conector que une los elementos **para** es una línea de puntos, lo cual indica que se trata de una conexión de contenido mixto.

4. Haga clic en el panel *Resultados* para ver el resultado de la asignación.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <Company-Person xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNames
3    <Name>Organization Chart</Name>
4    <Office>
5      <Name>Nanonull, Inc.</Name>
6      <Desc>
7        <para>The company was established in<bold> Vereno</bold>in 1995. Nanonull devel
8          </para>
9        <para>White papers and further information will be made available in the near future.
10       </para>
11     </Desc>
12   </Office>
13   <Office>
14     <Name>Nanonull Europe, AG</Name>
15     <Desc>
16       <para>In May 2000, Nanonull<italic>Europe</italic> was set up in Vienna. The team co
17     </Desc>
18   </Office>
19 </Company-Person>
20
```

5. Ahora haga clic en el icono **Ajuste automático de línea**  en la barra de iconos del panel *Resultados* para ver todo el texto.

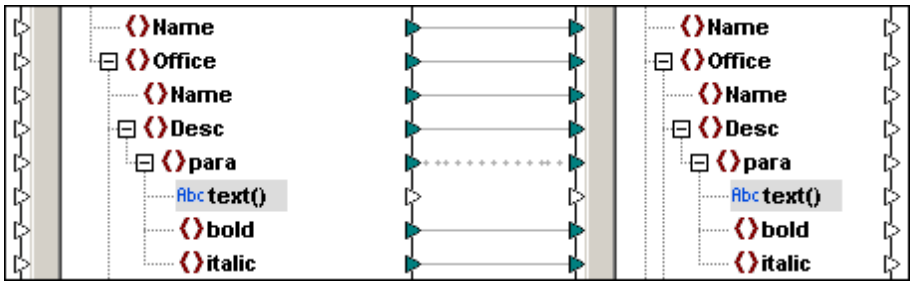
```
3     <Name>Organization Chart</Name>
4     <Office>
5       <Name>Nanonull, Inc.</Name>
6       <Desc>
7         <para>The company was established in<bold> Vereno</bold>in 1995. Nanonull develops
      nanoelectronic technologies for<italic>multi-core processors.</italic>February 1999 saw the
      unveiling of the first prototype <bold>Nano-grid.</bold>The company hopes to expand its
      operations <italic>offshore</italic>to drive down operational costs.
8           </para>
9           <para>White papers and further information will be made available in the near future.
10          </para>
11        </Desc>
12    </Office>
13    <Office>
14      <Name>Nanonull Europe, AG</Name>
15      <Desc>
16        <para>In May 2000, Nanonull<italic>Europe</italic> was set up in Vienna. The team
      consists of<bold> five research scientists </bold>and one administrative staff.</para>
17      </Desc>
18    </Office>
19  </Company-Person>
20
```

El texto de contenido mixto de cada descripción de las oficinas se asignó correctamente. El texto, además del contenido de las etiquetas `bold` e `italic` se asignó en el orden en el que aparecen en el archivo **XML** de origen.

6. Vuelva al panel *Resultados*.

Para quitar nodos de texto de elementos de contenido mixto:

1. Haga clic en el conector del nodo **text()** y pulse **Supr** para eliminarlo.



2. Ahora haga clic en el panel *Resultados* para ver el resultado de la asignación.

```
<Name>Nanonull, Inc.</Name>
<Desc>
<para>
<bold> Vereno</bold>
<italic>multi-core processors.</italic>
<bold>Nano-grid.</bold>
<italic>offshore</italic>
</para>
<para/>
</Desc>
</Office>
<Office>
<Name>Nanonull Europe, AG</Name>
<Desc>
<para>
<italic>Europe</italic>
<bold> five research scientists </bold>
</para>
</Desc>
```

Resultado:

- se eliminaron todos los nodos de **texto** del elemento `para`.
- se conservó el contenido de texto de los elementos `bold` e `italic` asignados.
- la **secuencia** de los elementos `bold` e `italic` sigue la secuencia del archivo XML de origen.

Para asignar instrucciones de procesamiento y comentarios:

1. Haga clic con el botón derecho en la conexión de contenido mixto y seleccione el comando **Propiedades**.
2. Bajo el grupo de opciones *Basada en origen (contenido mixto)* marque las casillas *Asignar instrucciones de procesamiento* y *Asignar comentarios*.

5.2.2.2 Ejemplo de contenido mixto

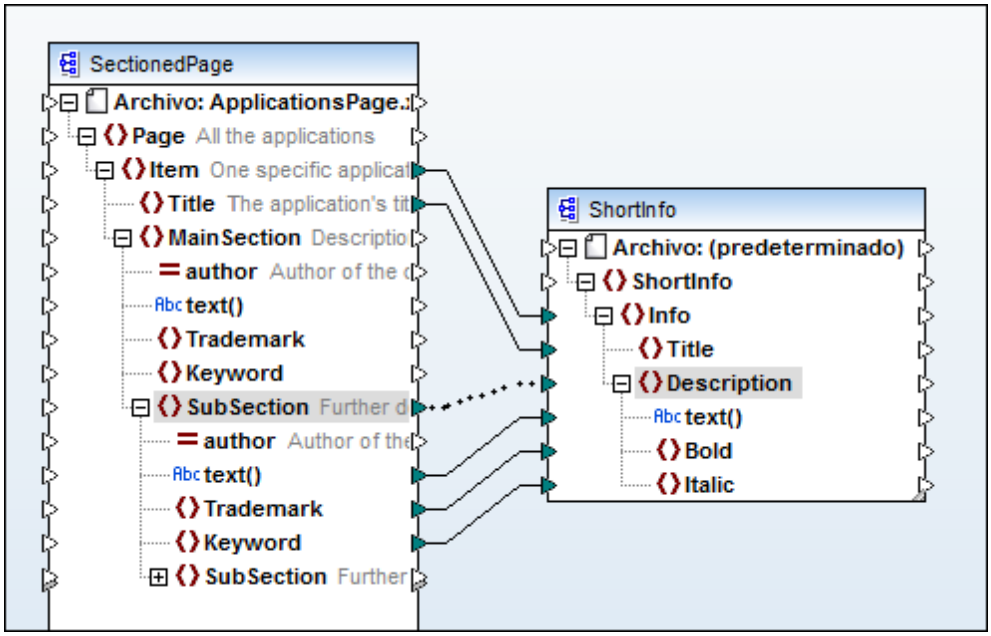
En este apartado usamos como ejemplo el archivo de asignación **ShortApplicationInfo.mfd**, disponible en la carpeta `...\MapForceExamples`.

Aquí puede ver un fragmento del archivo XML de origen de esta asignación:

```
<Page xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="SectionedPage.xsd">
    <Item>
        <Title>XMLSpy</Title>
        <MainSection author="altova">
            Altova <Trademark>XMLSpy</Trademark>
            <SubSection>Altova <Trademark>XMLSpy</Trademark> 2005 Enter
is the industry standard <Keyword>XML</Keyword> development environment
editing, debugging and transforming all <Keyword>XML</Keyword> technolo
automatically generating runtime code in multiple programming languages
        </MainSection>
    </Item>
```

Más abajo puede ver también una imagen de la asignación, en la que podrá observar que:

- El conector del nodo `SubSection` es de contenido mixto y está asignado al nodo `Description` del esquema/XML de destino.
- Los nodos `text()` están conectados.
- El texto de `Trademark` está asignado al nodo `Bold` del componente de destino.
- El texto de `Keyword` está asignado al nodo `Italic` del componente de destino.



Resultado de la asignación

El texto de contenido mixto de cada descripción se asignó correctamente. El texto, así como el contenido de las etiquetas `bold` e `italic`, se asignó tal y como aparecen en el archivo XML de origen.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <ShortInfo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:noNamespaceSchemaLocation="
   C:/PROGRA~1/Altova/MapForce2005/MapForceExamples/ShortInfo.xsd">
3    <Info>
4      <Title>XMLSpy</Title>
5      <Description>Altova <Bold>XMLSpy</Bold> 2005 Enterprise Edition is the industry standard
   <Italic>XML</Italic> development environment for modeling, editing, debugging and transforming
   all <Italic>XML</Italic> technologies, then automatically generating runtime code in multiple
   programming languages.</Description>
6    </Info>
```

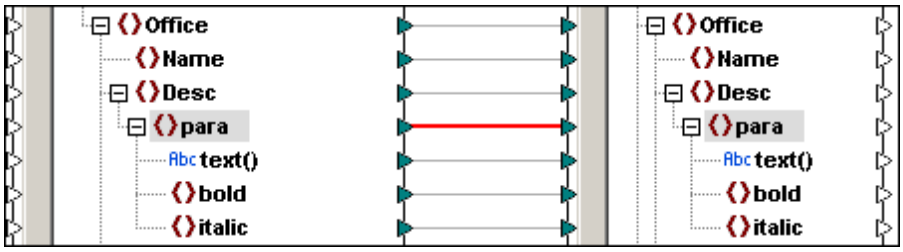
5.2.2.3 Usar conexiones estándar en nodos de contenido mixto

Como se dijo previamente, las conexiones basadas en el origen (o conexiones no estándar) suelen utilizarse para asignar datos de nodos de contenido mixto. De lo contrario, podrían obtenerse resultados no deseados. A continuación mostramos un ejemplo de lo que ocurre cuando se usa una conexión estándar (basada en el destino) para asignar datos de un nodo de contenido mixto:

1. Abra la asignación **Tut-OrgChart.mfd** de la carpeta **<Documentos>\Altova**

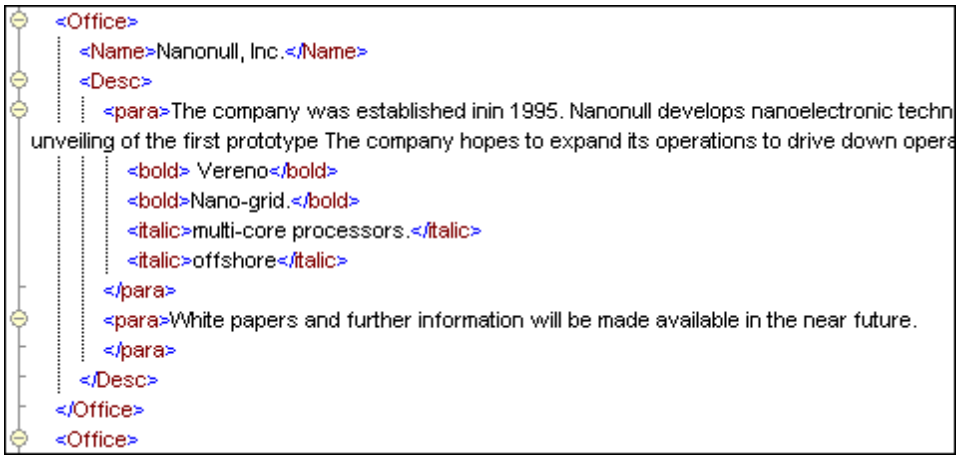
\MapForce2018\MapForceExamples\Tutorial.

2. Cree una conexión entre el nodo `para` del componente de origen y el nodo `para` del componente de destino. Aparece un mensaje preguntando si desea que MapForce defina las conexiones como basadas en el origen. Haga clic en **No** para ignorar la sugerencia de MapForce y crear una conexión estándar.



Nota: compruebe que la conexión es estándar (basada en el destino). Si se creó una conexión de copia total automáticamente, haga clic con el botón derecho en la conexión y seleccione **Basada en destino (estándar)** en el menú contextual.

3. Abra el panel *Resultados* para ver el resultado de la asignación.



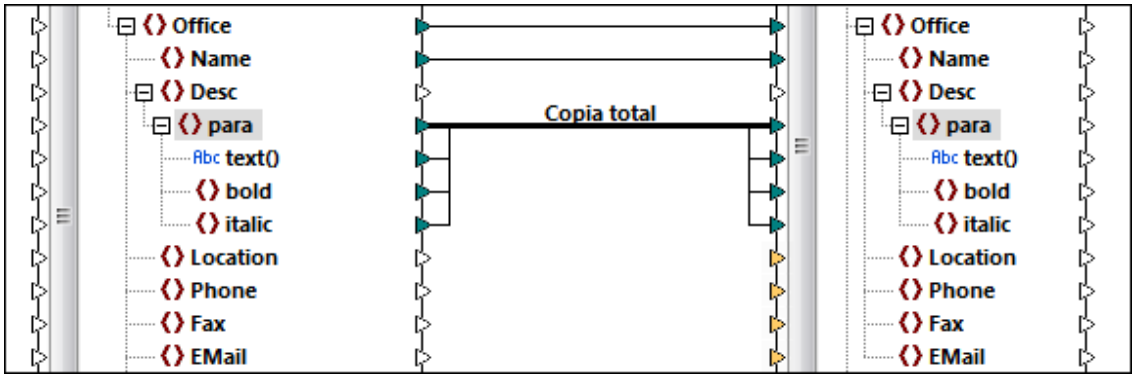
Como puede verse en la imagen, la asignación de nodos de contenido mixto por medio de conexiones estándar produce este resultado:

- El contenido del elemento de origen `text()` se copia en el destino, pero la secuencia de nodos secundarios (`bold` e `italic` en este caso) en el documento de salida corresponde a la secuencia de los nodos en el esquema XML de destino. En otras palabras, los nodos secundarios (`bold` e `italic` en este caso) aparecen después del texto del nodo de contenido mixto.
- Por cada elemento `para` MapForce asigna primero el nodo `text()`, después todos los elementos `bold` y, por último, todos los elementos `italic`. Como resultado aparecen varios elementos `bold` y `italic` apilados. Observe que el contenido de cada elemento se asigna solo si está conectado con el origen.

5.2.3 Conexiones de copia total

Las conexiones de copia total crean asignaciones de datos entre estructuras complejas (nodos con elementos secundarios) que son muy parecidas o idénticas. La ventaja principal de las conexiones de copia total es que simplifican el área de trabajo (se crea una conexión gruesa en lugar de muchas).

En la asignación las conexiones de copia total se representan por medio de una línea gruesa (con horquillas de entrada y salida para cada uno de los elementos secundarios) que conecta dos estructuras idénticas o similares.



Conexión de copia total

Cuando se dibuja una conexión entre dos estructuras en la asignación y MapForce detecta que la estructura de origen y la de destino son compatibles (es decir, que ambas son del mismo tipo que la de destino es un subtipo de la de origen), entonces se crea automáticamente una conexión de copia total. Cuando se ejecute la asignación, todos los datos de instancia se copiarán del origen al destino recursivamente, incluidos los datos secundarios.

Para crear conexiones de copia total a mano haga clic con el botón derecho en una conexión que conecte dos nodos similares y que tengan elementos secundarios y seleccione **Copia total (copia los elementos secundarios)** en el menú contextual.

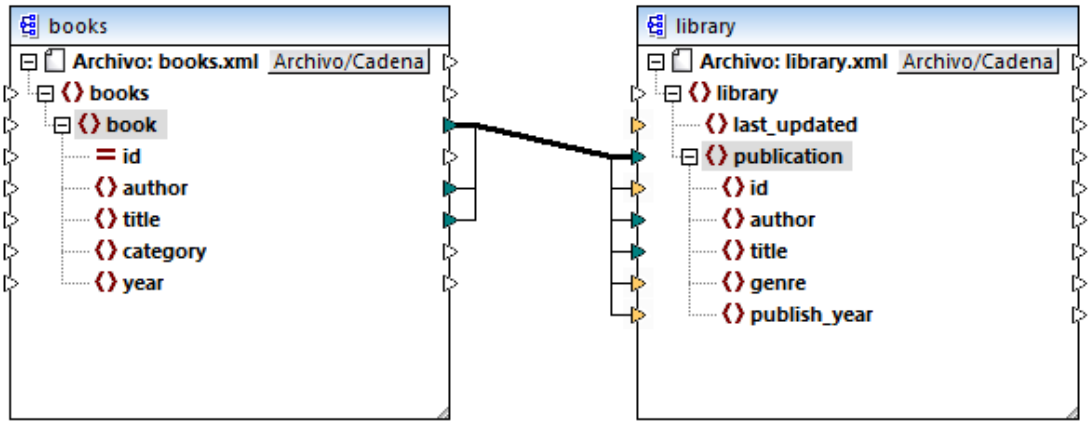
Debe tener en cuenta que:

- En contextos donde la conexión de copia total no es relevante o no es compatible, no se puede crear este tipo de conexión de forma manual.
- No se puede crear una conexión de copia total con el elemento raíz `root` de un componente XML o de esquema.
- Cuando se creen conexiones de copia total entre un esquema y un parámetro de una función definida por el usuario, los dos componentes deben estar basados en el mismo esquema (aunque no es necesario que tengan el mismo elemento raíz).

Para crear una conexión de copia total a mano:

1. Cree una asignación nueva.
2. Haga clic en el comando **Insertar | Archivo o esquema XML** y navegue hasta el archivo **books.xml** de la carpeta **<Documentos>\Altova\MapForce2018\MapForceExamples\Tutorial**.

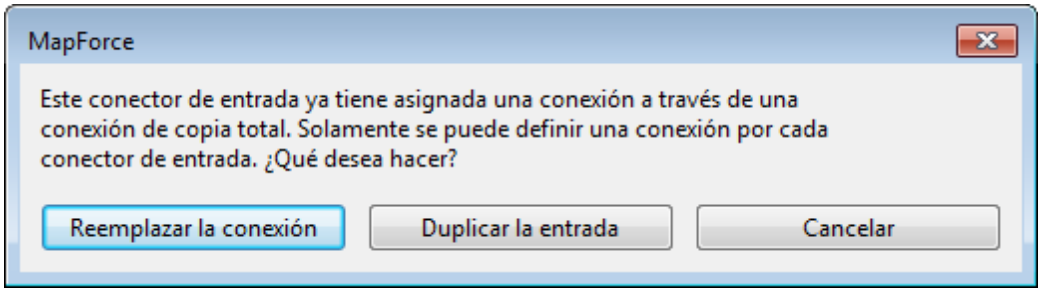
3. Haga clic en el comando **Insertar | Archivo o esquema XML** y navegue hasta el archivo **library.xsd** de la carpeta **<Documentos>\Altova\MapForce2018 \MapForceExamples\Tutorial**.
4. Dibuje una conexión de asignación entre el nodo `book` del componente "books" y el nodo `publication` del componente "library".
5. Haga clic con el botón derecho en la nueva conexión y seleccione **Copia total (copia los elementos secundarios)** en el menú contextual.



Si existen diferencias entre la estructura de origen y de destino, en tiempo de ejecución la conexión de copia total enumerará los nodos de origen (elementos y atributos) y solo copiará los que existan en el tipo de destino (esto se repite recursivamente).

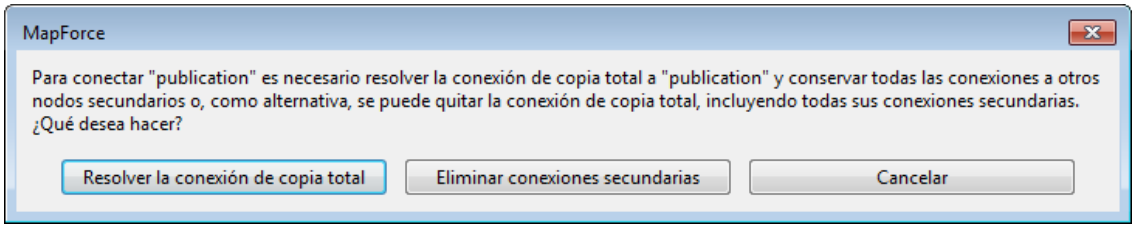
Por ejemplo, en la asignación de la imagen anterior, solamente hay dos elementos secundarios que son idénticos en ambas estructuras (`author` y `title`) y, por tanto, están conectados. El nodo `id` no se incluye automáticamente porque en el origen es un atributo y en el destino es un elemento. Si necesita asignar `category` a `genre`, por ejemplo, la conexión de copia total ya no es posible porque se trata de nodos distintos.

Cuando un conector de entrada (el icono en forma de triángulo situado en el lateral del componente) recibe una conexión de copia total, no podrá aceptar ninguna otra conexión. Por ejemplo, en la imagen anterior, si intentamos crear una conexión entre `category` y `genre`, MapForce solicitará que reemplace la conexión o que cree un duplicado de entrada.



Crear un duplicado de la entrada (*véase* *Duplicar entradas*) solo tiene sentido si el objetivo es que el destino acepte datos de más de una entrada (este no es el caso en este ejemplo). Si elige reemplazar la conexión de copia total, MapForce emite otro mensaje solicitando que se resuelva

o elimine la conexión de copia total.

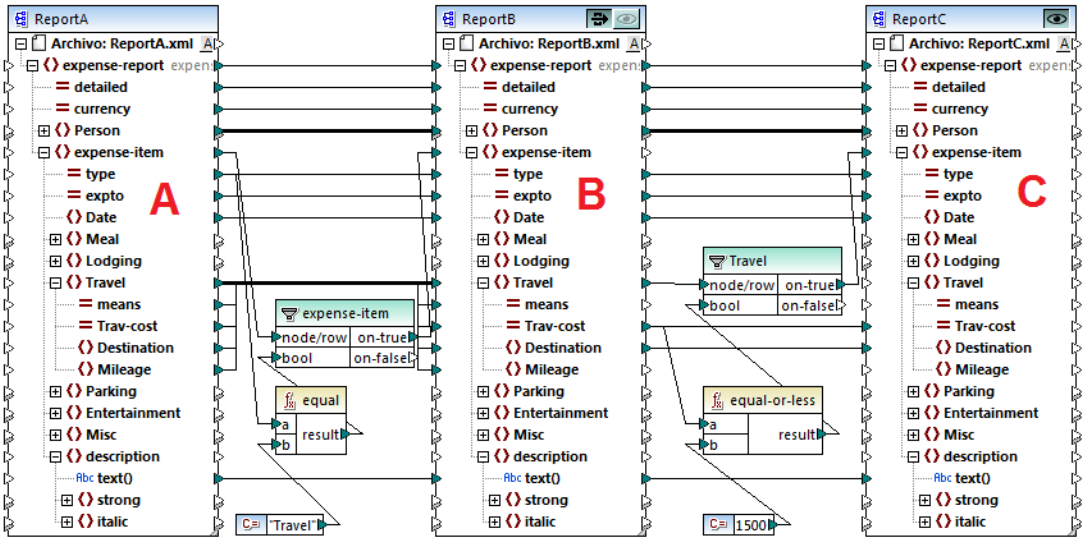


Haga clic en **Resolver la conexión de copia total** si desea reemplazar la conexión de copia total con varias conexiones estándar basadas en el destino. Si prefiere eliminar la conexión de copia total completamente, haga clic en **Eliminar conexiones secundarias**.

5.3 Asignaciones en cadena

En MapForce puede crear asignaciones de datos formadas por varios componentes en cadena. Las asignaciones en cadena son asignaciones donde como mínimo un componente hace de origen y de destino a la vez. Este tipo de componentes crea un resultado que se usa después como entrada del siguiente paso de la asignación en cadena. A estos componentes se les denomina *componente intermedio*.

Por ejemplo, la asignación que aparece a continuación muestra un informe de gastos (en formato XML) que se procesa en dos etapas. La parte de la asignación que va de A a B filtra los gastos marcados como "Travel" (viaje). La parte que va de B a C filtra los gastos de viaje por un importe inferior a 1500. El componente B es el componente intermedio porque tiene conexiones de entrada y salida. Puede encontrar esta asignación en esta ruta de acceso: **<Documentos>\Altova\MapForce2018\MapForceExamples\Tutorial\ChainedReports.mfd**.



ChainedReports.mfd

Las asignaciones en cadena presentan una característica llamada *paso a través.* Se trata de una función de vista previa que permite consultar los resultados que produce cada etapa de la asignación en cadena (en el panel *Resultados*). Por ejemplo, en la asignación del ejemplo puede consultar y guardar la vista previa del resultado XML que produce la etapa A - B y del resultado XML que produce la etapa B - C.

Nota: 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 pueden ser componentes intermedios pero no disponen del botón **Paso a través**. El componente intermedio siempre se regenera desde cero cuando se crea la vista previa o se genera código. Esto no sería posible con bases de datos porque antes de regenerar la base de datos sería necesario eliminarla.

Si la asignación se ejecuta con MapForce Server o con código generado, entonces se ejecuta

toda la cadena de asignación. La asignación genera los archivos de salida necesarios en cada etapa de la cadena y el resultado de una etapa se reenvía como entrada de la siguiente etapa.

También se pueden generar nombres de archivo dinámicos a partir de los componentes intermedios. Es decir, se pueden crear conexiones con el elemento `Archivo:` de los componentes intermedios, siempre y cuando éstos estén configurados correctamente (*véase* *Procesar varios archivos de entrada o salida simultáneamente*).

Botón Vista previa

Tanto el componente B como el componente C tienen un botón de vista previa. Este botón genera en MapForce una vista previa del resultado intermedio de B, así como el resultado final de la asignación en cadena. Haga clic en el botón **Vista previa** del componente correspondiente y después haga clic en el panel *Resultados* para ver el resultado de la asignación.

Los componentes intermedios que tengan activo el botón **Paso a través** no pueden verse en la vista previa porque su botón **Vista previa** se deshabilita automáticamente (ya que no tiene sentido obtener una vista previa de los datos y pasarlos al mismo tiempo). Para ver el resultado de los componentes intermedios, primero es necesario desactivar el botón **Paso a través** y después hacer clic en el botón **Vista previa**.

Botón Paso a través

El componente intermedio B tiene un botón más en la barra de título que se llama **Paso a través**.

Si el botón **Paso a través** está **activo** , MapForce asigna dirige todos los datos a la ventana de vista previa de una vez, del componente A al B y después al C. Se crearán dos archivos de resultados:

- el resultado de asignar el componente A al componente intermedio B.
- el resultado de asignar el componente intermedio B al componente de destino C.

Si el botón **Paso a través** está **inactivo** , MapForce solamente ejecutará partes de la cadena de asignación. Los datos que se generan dependen de qué botones **Vista previa** están activos:

- Si está activo el botón **Vista previa** del componente B, se genera el resultado de asignar el componente A al componente B. La cadena de asignación se detiene en el componente B. El componente C no se utiliza para nada en la vista previa.
- Si está activo el botón **Vista previa** del componente C, se genera el resultado de asignar el componente intermedio B al componente C. Como el paso a través está inactivo, se interrumpe el encadenamiento automático para el componente B. Solamente se ejecuta la parte derecha de la cadena de asignación. Es decir, el componente A no se utiliza.

> Cuando el botón Paso a través está inactivo, es importante que en los campos *Archivo XML de entrada* y *Archivo XML de salida* del componente intermedio se especifiquen nombres de archivo idénticos. Esto garantiza que el archivo de salida que se genera durante la vista previa de la etapa A - B sirva de entrada para la vista previa de la etapa B - C. Además, en el código generado o en la ejecución con MapForce Server, esto garantiza que la cadena de asignación no se rompa.

Como ya dijimos anteriormente, si la asignación se ejecuta con MapForce Server o con el código

generado, se da salida a todos los componentes. Cuando así sea, el estado del botón **Paso a través** del componente B no se tiene en cuenta (tampoco se tiene en cuenta qué componente de visa previa está seleccionado). Por ejemplo, a partir de la asignación de la imagen se generarían dos archivos de resultados:

1. el archivo de salida que se obtiene de la asignación de A a B y
2. el archivo de salida que se obtiene de la asignación de B a C.

Los apartados Ejemplo: paso a través activo y Ejemplo: paso a través inactivo explican con más detalle cómo se transfieren los datos de origen cuando el botón **Paso a través** está activo e inactivo, respectivamente.

5.3.1 Ejemplo: paso a través activo

La asignación utilizada en este ejemplo se llama **ChainedReports.mfd** y está en la carpeta **<Documentos>\Altova\MapForce2018\MapForceExamples\Tutorial**. Esta asignación procesa un archivo XML llamado **ReportA.xml** que contiene gastos de viaje (ver más abajo). Para mayor claridad se omitieron la declaración de espacio de nombres y algunos elementos `expense-item`:

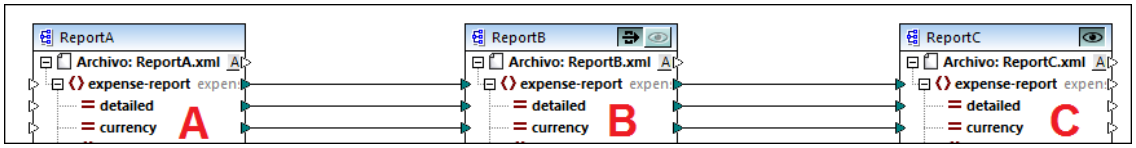
```
<?xml version="1.0" encoding="UTF-8"?>
<expense-report currency="USD" detailed="true">
   <Person>
      <First>Fred</First>
      <Last>Landis</Last>
      <Title>Project Manager</Title>
      <Phone>123-456-78</Phone>
      <Email>f.landis@nanonull.com</Email>
   </Person>
   <expense-item type="Travel" expto="Development">
      <Date>2003-01-02</Date>
      <Travel Trav-cost="337.88">
         <Destination/>
      </Travel>
      <description>Biz jet</description>
   </expense-item>
   <expense-item type="Lodging" expto="Sales">
      <Date>2003-01-01</Date>
      <Lodging Lodge-cost="121.2">
         <Location/>
      </Lodging>
      <description>Motel mania</description>
   </expense-item>
   <expense-item type="Travel" expto="Marketing">
      <Date>2003-02-02</Date>
      <Travel Trav-cost="2000">
         <Destination/>
      </Travel>
      <description>Hong Kong</description>
   </expense-item>
</expense-report>
```

Archivo ReportA.xml

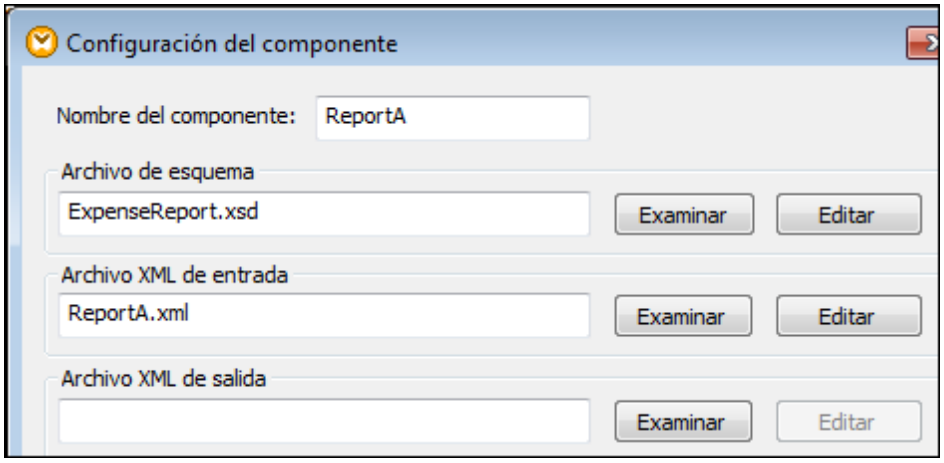
El objetivo de la asignación es producir dos informes a partir del archivo XML:

- El informe **ReportB.xml**, que debe contener los gastos de viaje que correspondan al tipo "Travel".
- El informe **ReportC.xml**, que debe contener los gastos de viaje que correspondan al tipo "Travel" y que no superen un valor de 1500.

Para conseguirlo el componente intermedio de la asignación (el componente B) tiene activado el botón **Paso a través** (*ver más abajo*). Esto hace que la asignación se ejecute en dos etapas: de A a B y de B a C. El resultado que genera el componente intermedio se usará como entrada para la asignación entre B y C.



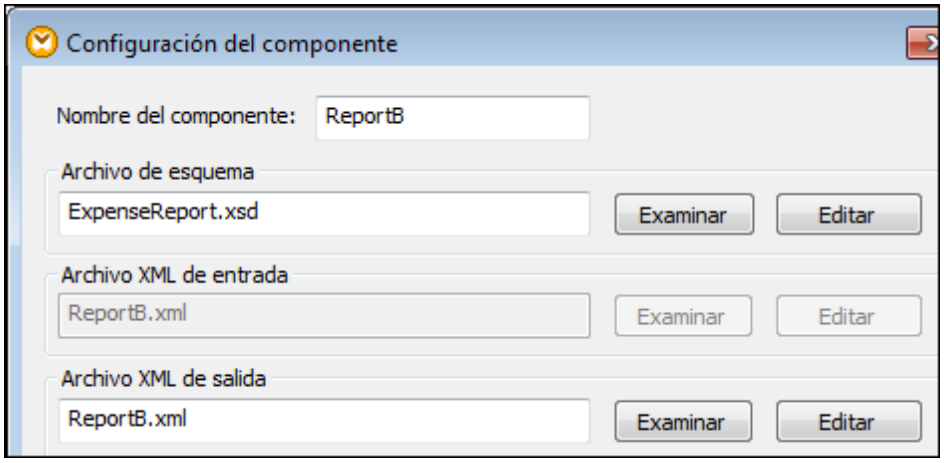
El nombre de los archivos de salida que se generan en cada etapa de la cadena de asignación viene dado por la configuración de cada componente. Para abrir la configuración de un componente haga clic con el botón derecho en el componente y seleccione **Propiedades** en el menú contextual. En nuestro ejemplo el primer componente está configurado para leer datos de un archivo XML llamado **ReportA.xml**. Como este es el componente de origen, el archivo de salida XML no es relevante y este campo se dejó vacío en la configuración del componente.



Configuración del componente de origen

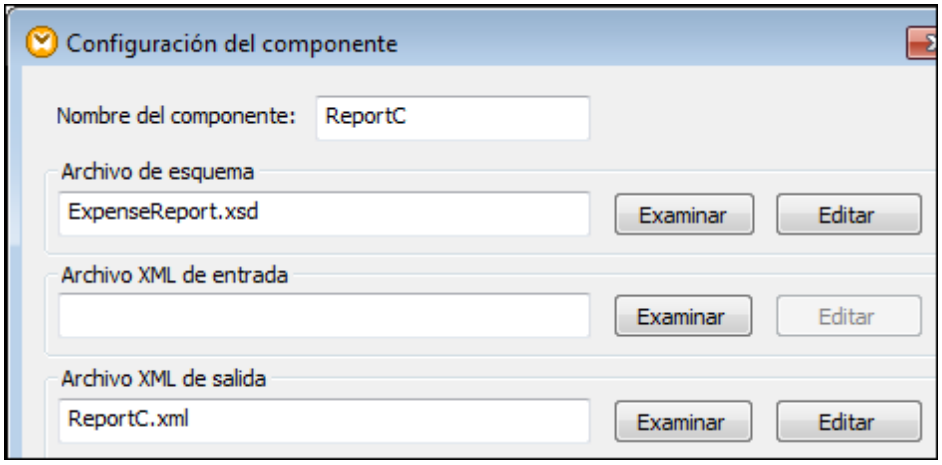
Como puede verse en la siguiente imagen, el segundo componente (**ReportB**) está configurado para generar un archivo de salida llamado **ReportB.xml**. Observe que el campo *Archivo XML de entrada* está atenuado. Esto se debe a que cuando el paso a través está activo (como en nuestro ejemplo) el campo *Archivo XML de entrada* del componente intermedio se desactiva automáticamente. Para poder ejecutar la asignación no se necesita un nombre de archivo de entrada porque el resultado que se crea en esta etapa de la cadena se almacena en un archivo temporal y se vuelve a utilizar en las etapas siguientes. Además, si se define un archivo XML de salida (*imagen siguiente*), éste se usará para el nombre de archivo del archivo de salida

intermedio. Si no se define el valor del campo *Archivo XML de salida*, se usará automáticamente un nombre de archivo predeterminado.



Configuración del componente intermedio

Por último, el tercer componente está configurado para generar un archivo de salida llamado **ReportC.xml**. El valor del campo *Archivo XML de entrada* no es relevante porque se trata del componente de destino.

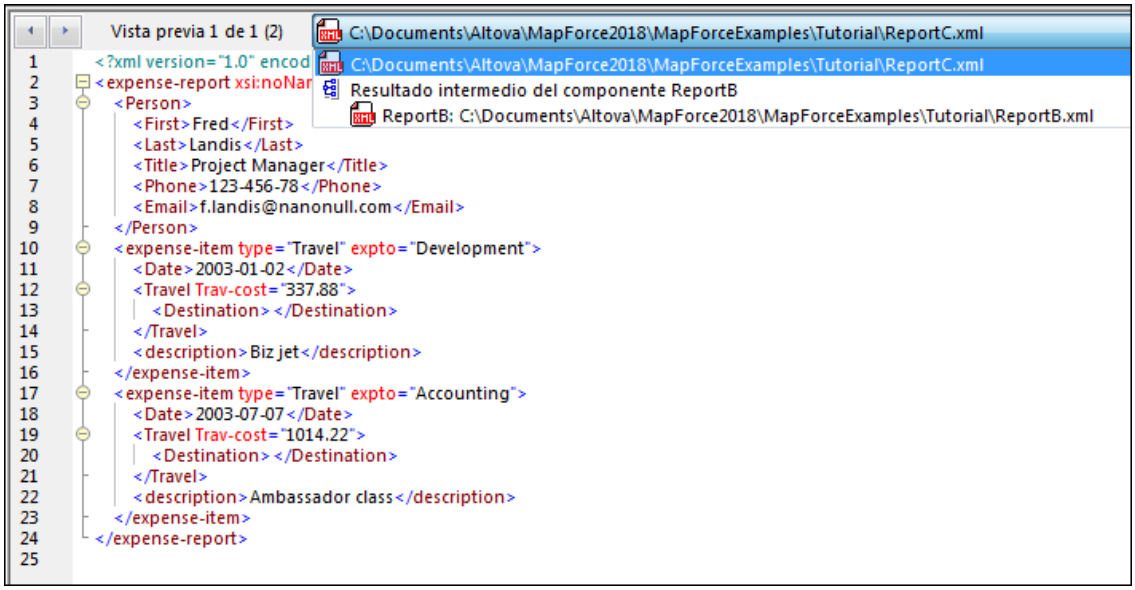


Configuración del componente de destino

Si consultamos la vista previa abriendo el panel *Resultados*, observaremos que se generaron dos archivos de salida:

1. **ReportB.xml**, que representa el resultado de la asignación de A a B.
2. **ReportC.xml**, que representa el resultado de la asignación de B a C.

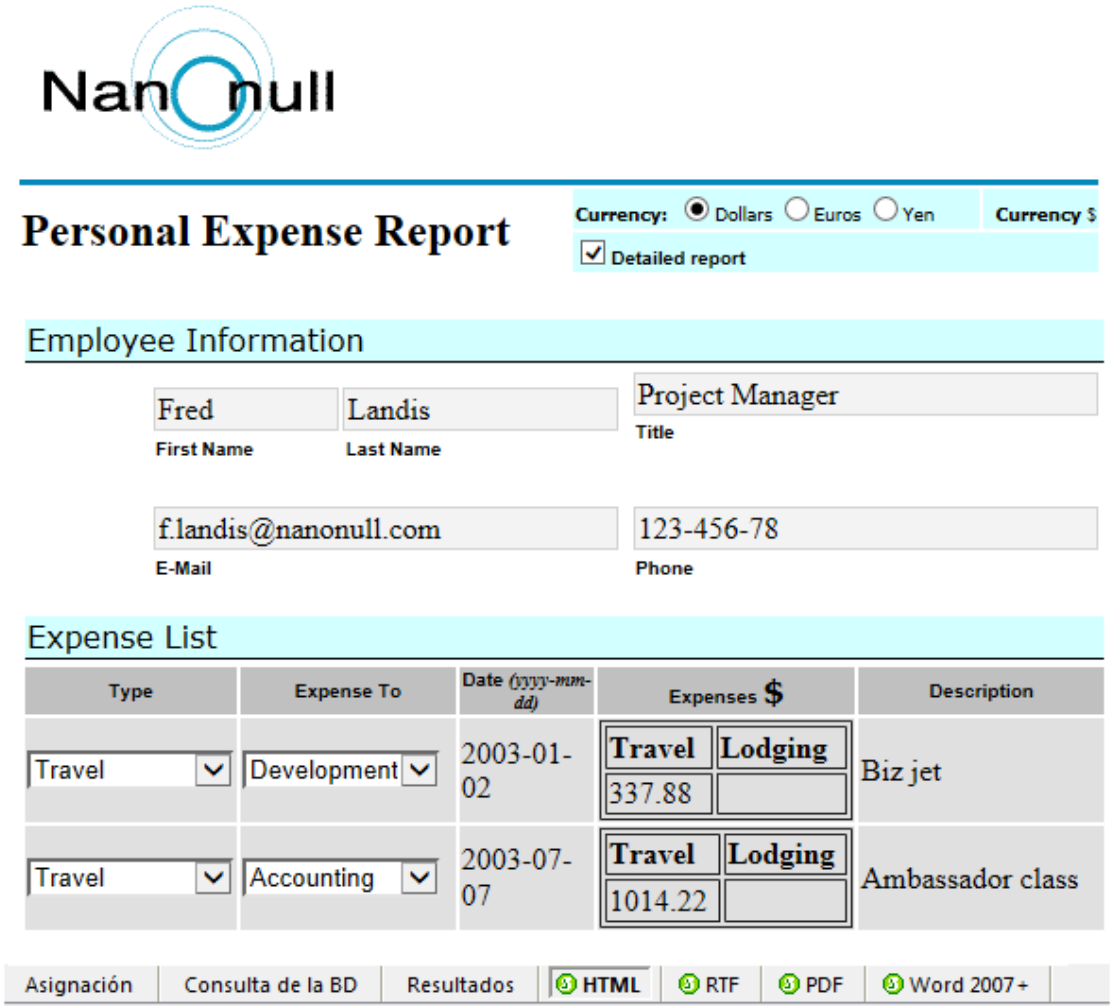
Para seleccionar cuál de los dos archivos de salida se presentan en el panel *Resultados* haga clic en los botones de flecha o seleccione una entrada en la lista desplegable.



Archivos de salida generados

Cuando MapForce ejecuta la asignación, la configuración de la opción *Escribir directamente en archivos de salida finales* (que se configura desde **Herramientas | Opciones | Generales**) determina si los archivos intermedios 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 ejecutara con MapForce o por medio de código, se producirían archivos en cada etapa de la cadena de asignación.

Si tiene StyleVision instalado y asignó un archivo SPS (StyleVision Power Stylesheet) al componente de destino (como en nuestro ejemplo), entonces podrá ver (y guardar) el resultado final de la asignación como HTML, RTF. Para generar y consultar este resultado en MapForce basta con abrir el panel correspondiente.

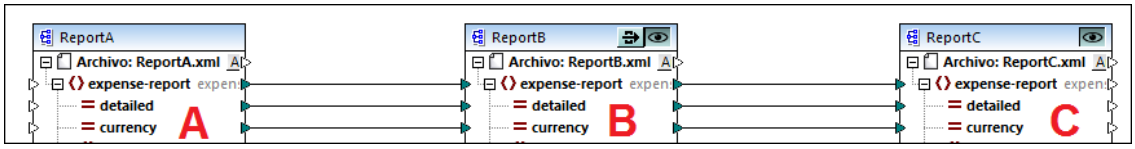


Resultado HTML generado

Recuerde que solamente se presenta el resultado del componente de destino final de la cadena de asignación. Para ver el resultado de StyleVision de componentes intermedios deberá desactivar el botón de paso a través y generar una vista previa del componente intermedio (*véase* *Ejemplo: paso a través inactivo*).

5.3.2 Ejemplo: paso a través inactivo

La asignación utilizada en este ejemplo se llama **ChainedReports.mfd** y está en la carpeta **<Documentos>\Altova\MapForce2018\MapForceExamples\Tutorial**. Este ejemplo muestra el resultado que genera la asignación cuando el botón **Paso a través** del componente intermedio está desactivado.

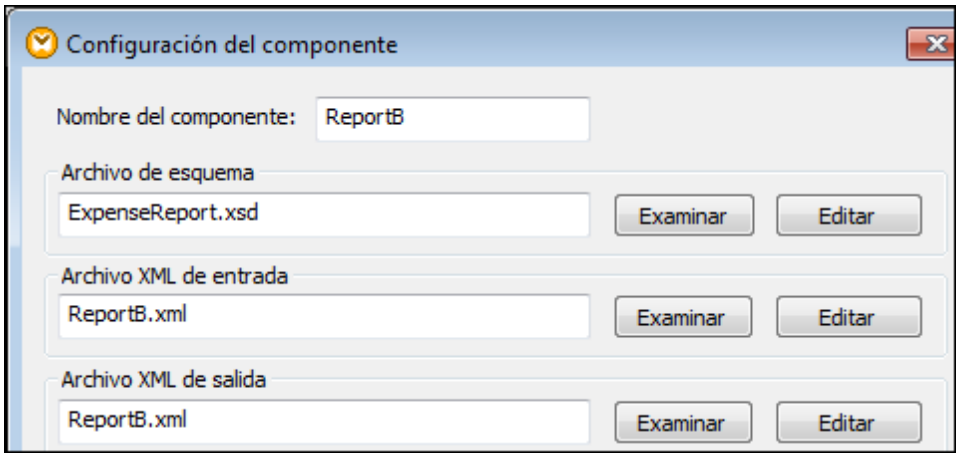


Como explicamos anteriormente en el apartado Ejemplo: paso a través activo, el objetivo de la asignación es producir dos informes distintos. En el ejemplo anterior, el botón **Paso a través** estaba activo y los informes generados podían verse en el panel *Resultados*. Sin embargo, si quiere consultar la vista previa de un solo informe (de **ReportB.xml** o de **ReportC.xml**), entonces debe desactivar el botón **Paso a través**. Desactivar el botón de paso a través puede ser de utilidad si su objetivo es:

- Consultar la vista previa del resultado que genera la asignación A - B solamente, ignorando la etapa B - C de la asignación.
- Consultar la vista previa del resultado que genera la asignación B - C solamente, ignorando la etapa A - B de la asignación.

Una vez desactivado el botón de paso a través, podrá elegir si consulta la vista previa de **ReportB** o de **ReportC** (observe que ambos tienen activo el botón **Vista previa**).

Además, desactivar el botón de paso a través permite 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* (como en este ejemplo).



Configuración del componente intermedio

El que el componente intermedio tenga el mismo archivo de entrada que de salida es importante si tiene pensado generar código a partir de la asignación o ejecutar la asignación con MapForce Server. Como se dijo anteriormente, en estos entornos se generan todos los resultados que crean los componentes de la cadena de asignación. Por tanto, lo más razonable es que el componente intermedio reciba un archivo (en este caso **ReportB.xml**), lo procese y lo reenvíe a la siguiente asignación, en lugar de buscar un nombre de archivo distinto. Tenga en cuenta que si en el componente intermedio el archivo de entrada y el de salida tienen nombres distintos (con el botón de paso a través inactivo), pueden producirse errores tipo *El sistema no encontró el archivo especificado* en el código generado o en la ejecución con MapForce Server.

Si hace clic en el botón **Vista previa**  del tercer componente (**ReportC**) e intenta generar la vista previa de resultados en MapForce, se producirá un error de ejecución. Se trata de un error esperado porque, según la configuración, se espera como entrada un archivo llamado **ReportB.xml** pero la asignación todavía no ha producido dicho archivo (porque el botón de paso a través no está activo y solo se ejecuta la etapa B - C de la asignación). Para solucionar este problema:

1. Haga clic en el botón de vista previa del componente intermedio.
2. Haga clic en el panel *Resultados* para generar una vista previa de los resultados.
3. Guarde el archivo de salida resultante como **ReportB.xml** en la carpeta donde está la asignación (**<Documentos>\Altova\MapForce2018\MapForceExamples\Tutorial**).

Si ahora vuelve a hacer clic en el botón de vista previa del tercer componente (**ReportC**), el error ya no se produce.

Desactivar el botón de paso a través también permite consultar la vista previa de resultados generados con StyleVision creados a partir de todos los componentes que tengan un archivo SPS (StyleVision Power StyleSheet) asociado. También podrá ver la versión HTML del informe intermedio (además del informe final):

Resultado HTML del componente intermedio

5.4 Procesar varios archivos de entrada o salida simultáneamente

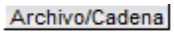
MapForce puede configurarse para procesar varios archivos simultáneamente (p. ej. todos los archivos de un directorio) cuando se ejecute la asignación. Gracias a esta característica podrá:

- 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:

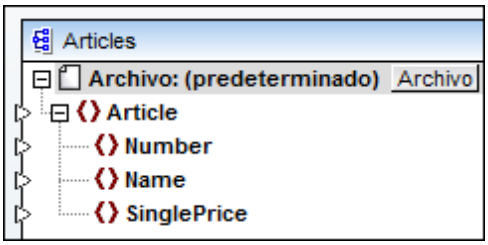
- Suministrando la ruta de acceso de los archivos de entrada/salida pertinentes por medio de caracteres de comodín (en lugar de dar un nombre de archivo fijo) en la configuración del componente (*véase* *Cambiar configuración de los componentes*). Es decir, puede introducir los comodines * y ? en el 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.

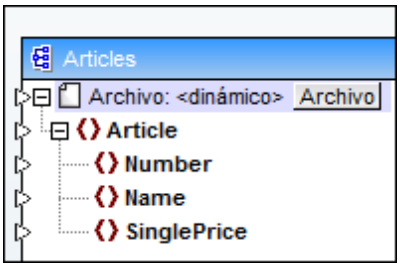
Dependiendo de cual sea el objetivo del proyecto, podrá usar una de estas técnicas o ambas en la misma asignación. Sin embargo, no tiene sentido utilizar ambas técnicas al mismo tiempo en el mismo componente. Para indicar qué técnica desea usar para cada componente haga clic en el botón 

o 

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**

** 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)	*
C++	*	*	*
C#	*	*	*
Java	*	*	*
BUILT-IN (motor integrado)	*	*	*

Leyenda:

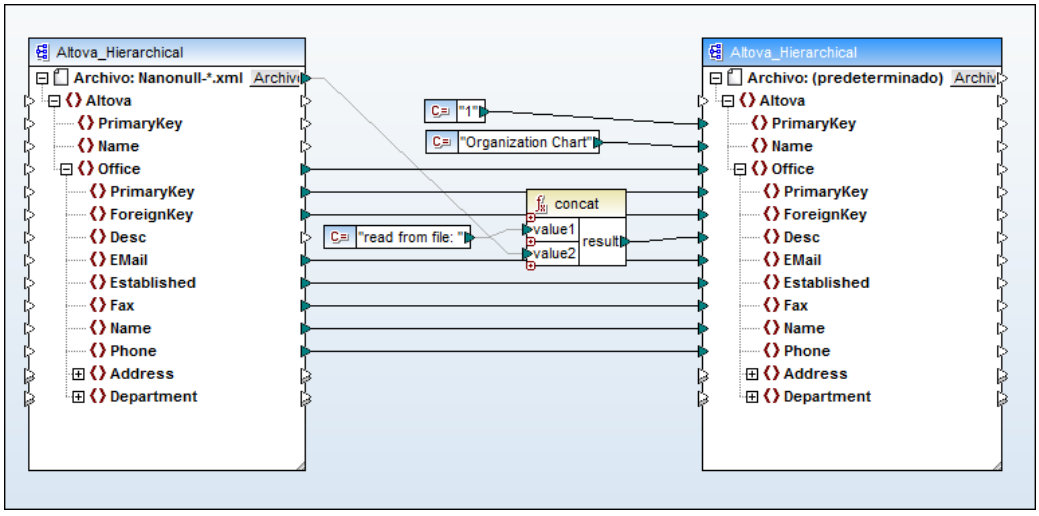
*	Compatible

(1)	Utiliza la función **`fn:collection`**. La implementación en los motores XSLT 2.0 y XQuery de Altova resuelve comodines. Los demás motores pueden comportarse de modo distinto. Para más información sobre transformación de código XSLT 1.0/2.0 con el motor RaptorXML Server, consulte los apartados Generar código XSLT

5.4.1 Asignar varios archivos de entrada a un solo archivo de salida

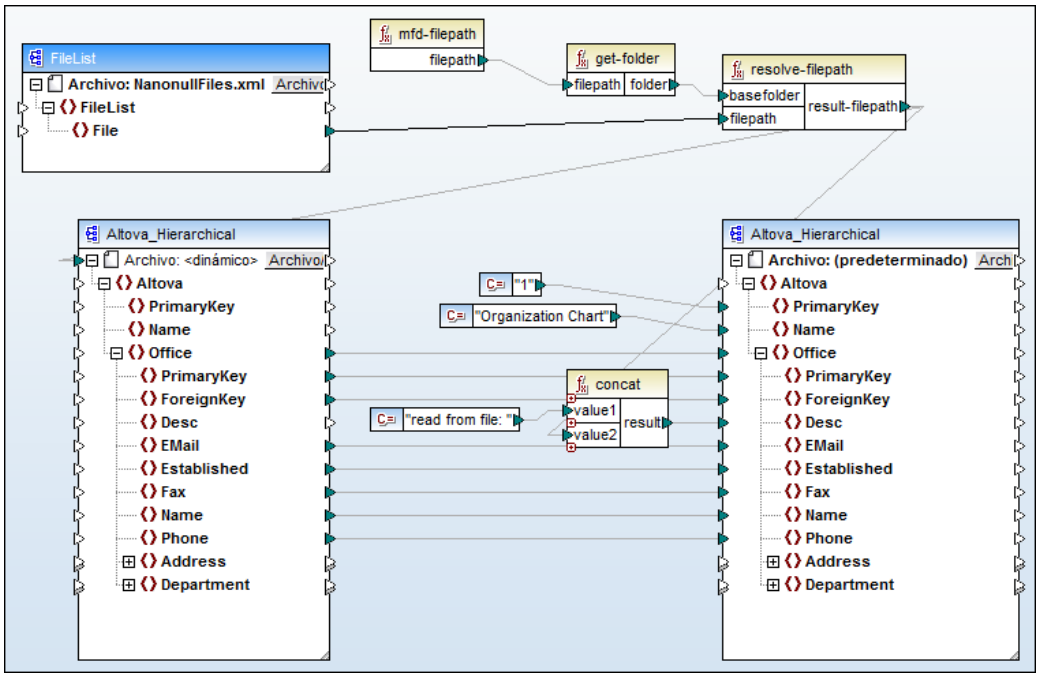
Hay dos maneras de procesar varios archivos de entrada a la vez:

- Introduciendo una ruta de acceso con comodines (* o ?) como archivo de entrada en el cuadro de diálogo "Configuración del componente". Se procesarán todos los archivos que coincidan con la ruta de acceso indicada. En la imagen siguiente, por ejemplo, en el campo *Archivo XML de entrada* se usa el comodín * para indicar que todos los archivos cuyo nombre empiece por `Nanonull-` serán los archivos de entrada de la asignación. En este ejemplo se combinan varios archivos de entrada para crear **un solo archivo de salida**. Como se puede apreciar en la imagen, no hay ningún conector dinámico con el componente de destino y el componente de origen tiene acceso a varios archivos gracias al comodín *. Observe también que el nombre del nodo raíz del componente de destino es **`Archivo: <predeterminado>`**. Es decir, no se definió ninguna ruta de acceso de archivo de salida en el cuadro de diálogo "Configuración del componente". Por tanto, al documento de destino se anexan varios archivos de origen.



MergeMultipleFiles.mfd (MapForce Basic Edition)

- Asignando una **secuencia** de cadenas al nodo **`Archivo`** del componente de origen. Cada cadena de la secuencia representa un nombre de archivo. Las cadenas también pueden contener comodines (que se resuelven automáticamente). Los componentes que pueden suministrar secuencias de nombres de archivos son los archivos XML.

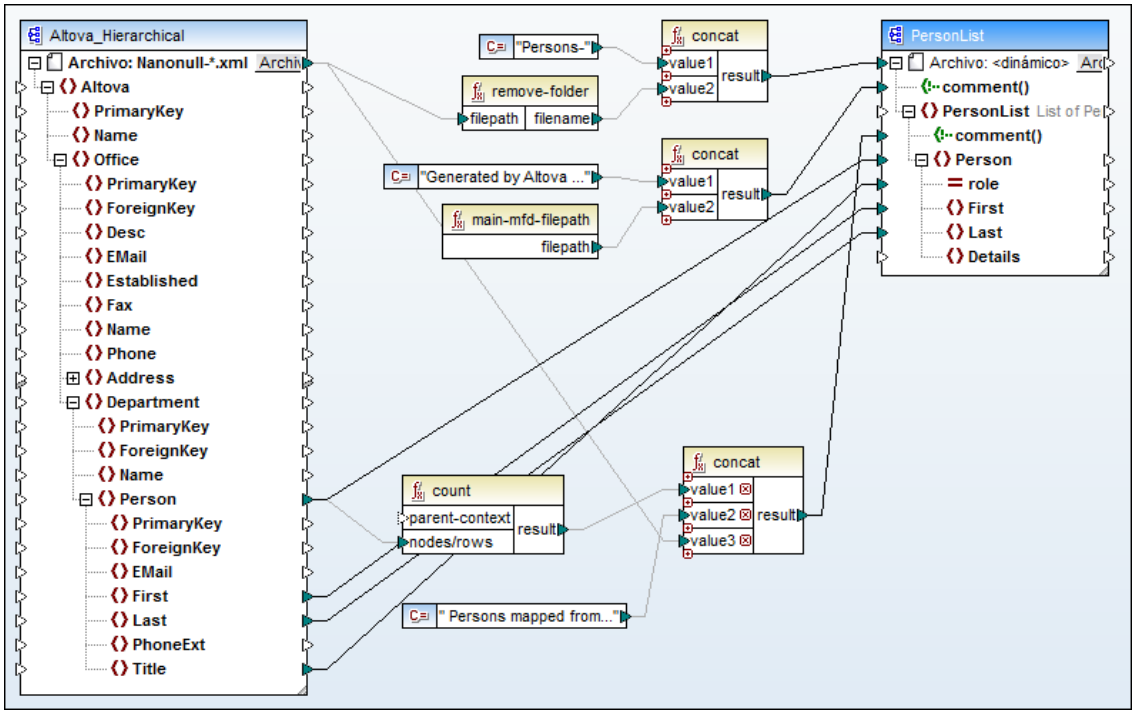


MergeMultipleFiles_List.mfd (MapForce Basic Edition)

5.4.2 Asignar varios archivos de entrada a varios archivos de salida

Para asignar varios archivos a varios archivos de destino es necesario generar nombres de archivo de salida únicos. En algunos casos, el nombre de los archivos de salida se puede derivar de cadenas existentes en los datos de entrada. En otros casos, lo más práctico es derivar el nombre de los archivos de salida del nombre del archivo de entrada (p. ej. cambiando la extensión de archivo).

En la asignación que aparece a continuación el nombre del archivo de salida se deriva del nombre del archivo de entrada: se añadió el prefijo **`Persons-`** con ayuda de la función **`concat`**.



MultipleInputToMultipleOutputFiles.mfd (MapForce Basic Edition)

Nota: recomendamos conectar simplemente los nodos raíz de entrada y salida directamente, sin usar funciones de procesamiento. Esto sobrescribiría los archivos de entrada cuando se ejecute la asignación. Puede cambiar los nombres de los archivos de salida con ayuda de funciones como la función **concat**.

Con el comando de menú **Archivo | Configurar asignación** puede definir la configuración de rutas de acceso global que debe utilizar la asignación (véase Cambiar configuración de la asignación).

5.4.3 Especificar nombres de archivo como parámetros de asignación

Para indicar nombres de archivo personalizados como parámetros de entrada de la asignación:

1. Añada un componente de entrada a la asignación (**Función | Insertar componente de entrada**). Para más información consulte el apartado Agregar componentes de entrada simples.
2. Haga clic en el botón Archivo o Archivo/Cadena del componente de origen y seleccione **Usar nombres de archivo dinámicos dados por la asignación** en el menú contextual.
3. Conecte el componente de entrada **input** con el nodo raíz del componente de origen.

Para ver un ejemplo más detallado consulte el apartado Ejemplo: usar nombres de archivo como parámetros de asignación.

5.4.4 Vista previa de varios archivos de salida

Haga clic en el panel *Resultados* para ver el resultado de la asignación. Si la asignación produce varios archivos de salida, cada archivo contará con un panel numerado en el panel *Resultados*. Haga clic en los botones en forma de flecha para navegar por los distintos archivos de salida.

MultipleInputToMultipleOutputFiles.mfd

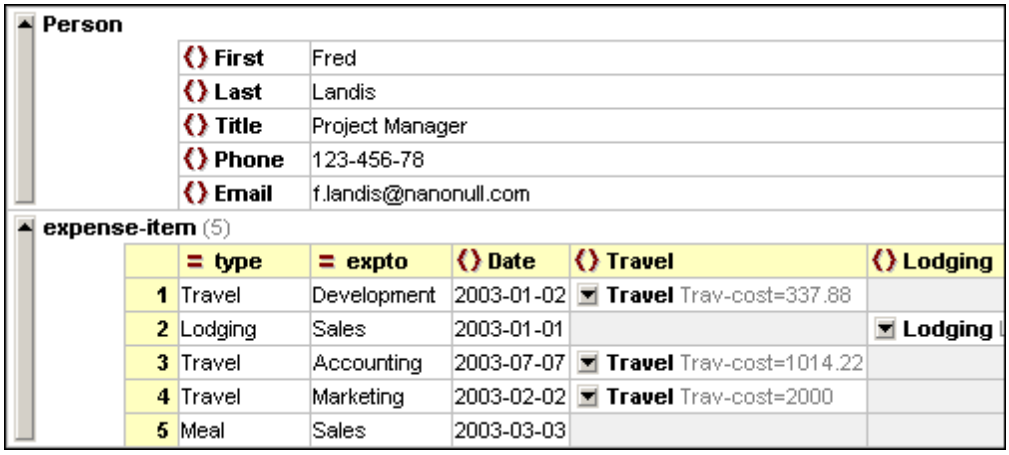
Hay dos maneras de guardar los archivos de salida generados:

- En el menú **Resultados** haga clic en el comando **Guardar todos los archivos de salida** ( ).
- Haga clic en el botón **Guardar todos** ( ) de la barra de herramientas.

5.4.5 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\MapForce2018\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

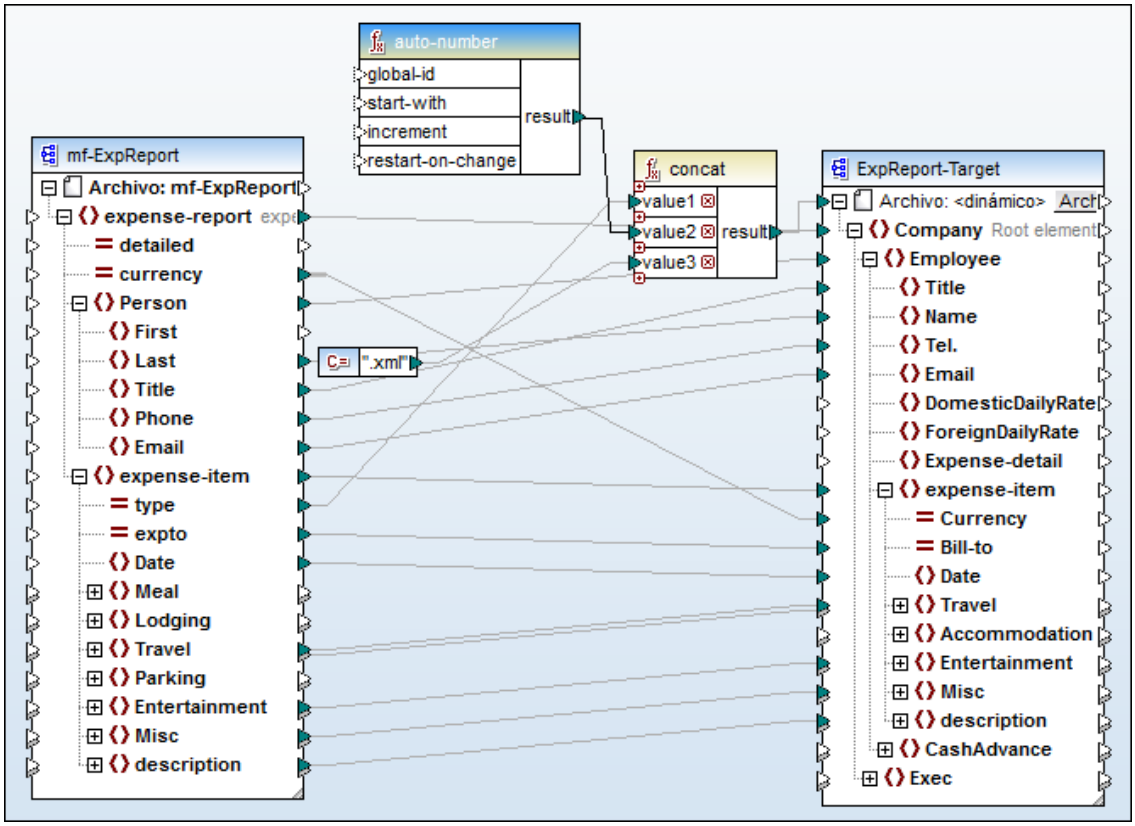
expense-item (5)

	= 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".

5.5 Pasar parámetros a la asignación

Puede pasar valores simples a una asignación por medio de componentes de entrada simples. En el área de asignación los componentes de entrada simple desempeñan el papel de componente de origen que tiene un tipo de datos simple (p. ej. cadena, entero, etc.) en lugar de una estructura de elementos y secuencias. Por consiguiente, puede crear un componente de entrada simple en lugar de (o además de) un componente de origen basado en un archivo.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*). Si fuera necesario, también podrá crear valores predeterminados para los parámetros de entrada de la asignación (*véase Crear un valor de entrada predeterminado*). 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*). 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 solo 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.

Puede consultar un ejemplo en el apartado Ejemplo: usar nombres de archivo como parámetros de asignación.

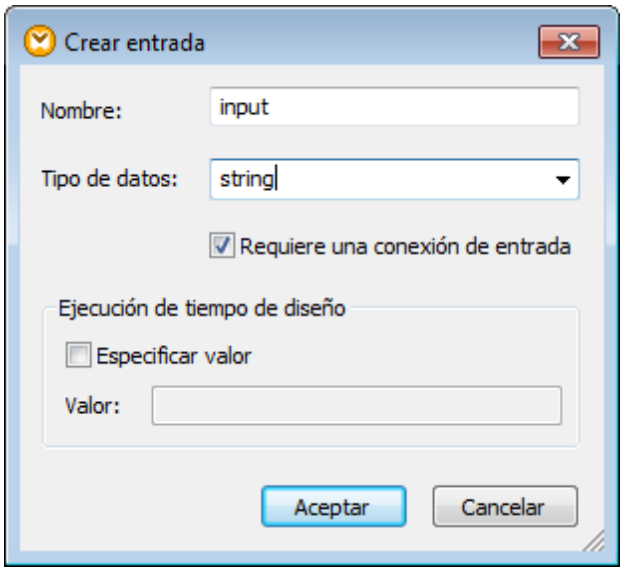
5.5.1 Agregar componentes de entrada simples

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. En el menú **Función** haga clic en **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.)

4. Haga clic en **Aceptar**.

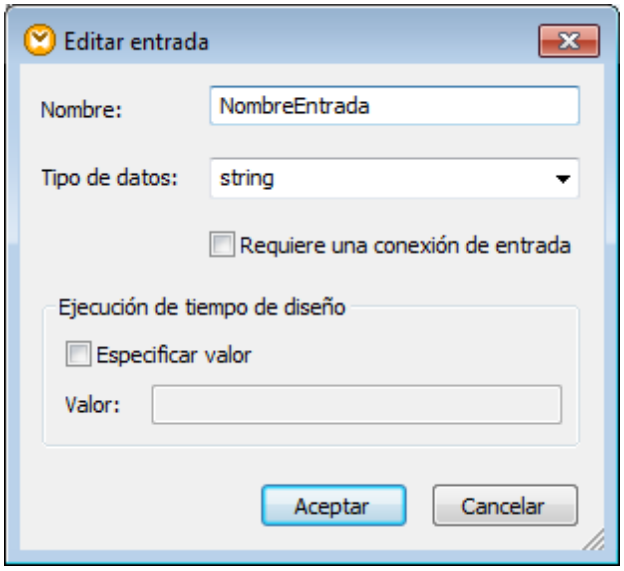


Cuadro de diálogo "Crear entrada"

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

5.5.2 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" o "Editar entrada" respectivamente.



Cuadro de diálogo "EdEdit Input dialog box

Para abrir el cuadro de diálogo "Editar 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.

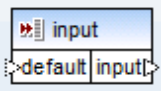
Estas son las opciones de configuración que ofrece el cuadro de diálogo:

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*).
Especificar valor	Esta casilla solo 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 solo 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.

5.5.3 Crear un valor de entrada predeterminado

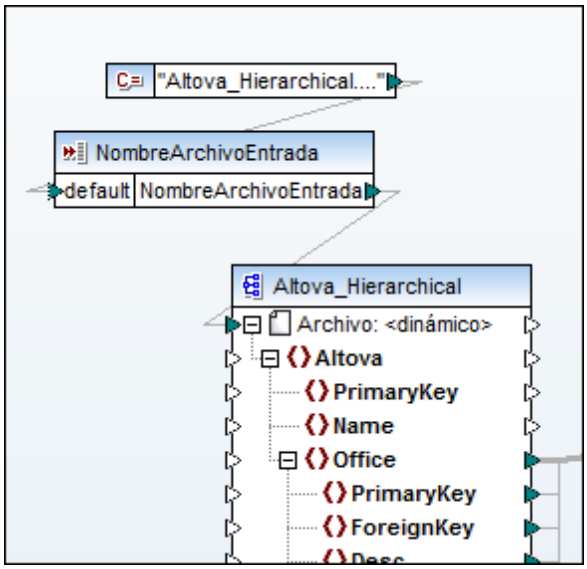
Tras añadir el componente de entrada al área de asignación, debemos fijarnos en el elemento **`default`** situado en el lado izquierdo del componente.



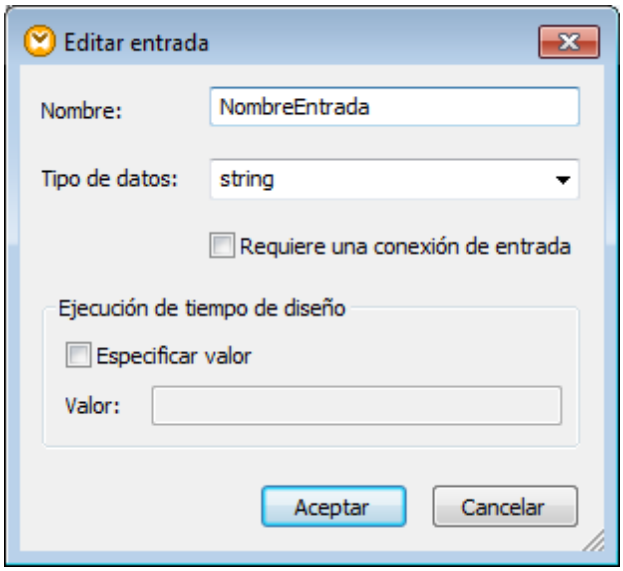
Componente de entrada simples

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.



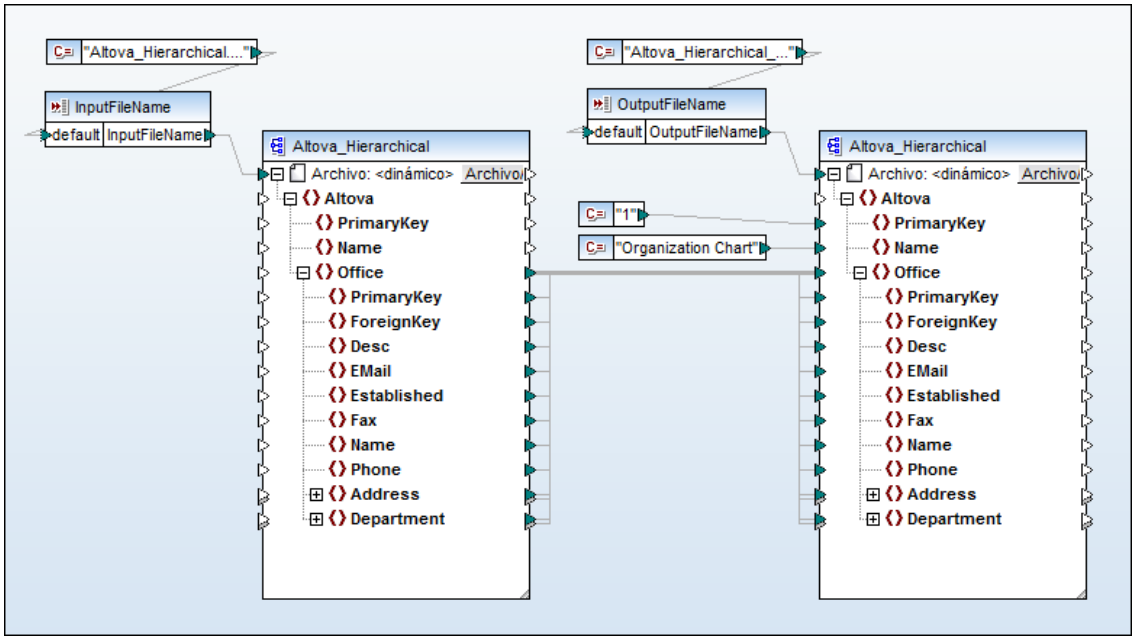
3. Haga clic en **Aceptar** para terminar.

Nota: si marca la casilla *Especificar valor* e introduce un valor en el campo adyacente, dicho valor tendrá prioridad sobre el valor predeterminado cuando consulte la vista previa de la asignación (es decir, cuando ejecute la asignación en tiempo de diseño). Sin embargo, ese valor no tendrá efecto alguno en el código generado.

5.5.4 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\MapForce2018\MapForceExamples\FileNamesAsParameters.mfd`.

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>`.



FileNamesAsParameters.mfd (MapForce Basic Edition)

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, con ayuda de RaptorXML Server

XSLT 2.0

Si genera código en XSLT 1.0 o XSLT 2.0, los parámetros de entrada se escriben en el archivo por lotes `DoTransform.bat` que se ejecutará con RaptorXML Server (*véase Automatización con RaptorXML Server*). 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`:

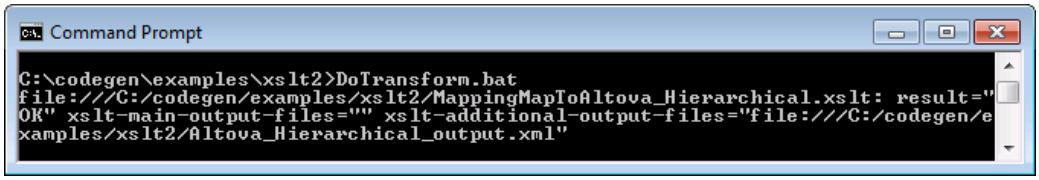
1. Genere código XSLT 2.0 (**Archivo | Generar código en | XSLT 2.0**) a partir del diseño de asignación `FileNamesAsParameters.mfd`.

2. Copie el archivo `Altova_Hierarchical.xml` del directorio `<Documentos>\Altova \MapForce2018\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.



5.6 Obtener valores de cadena de una asignación

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*).

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*). 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 solo 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	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

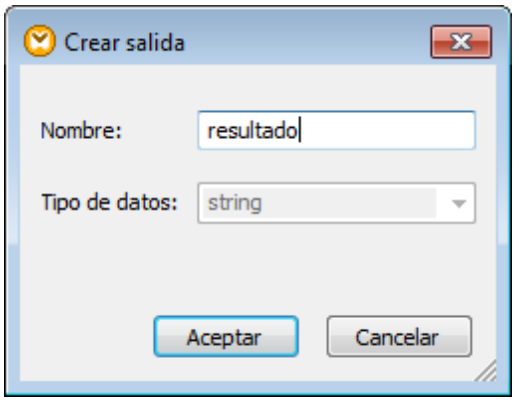
	comando siguiente se editó para que se escriba el resultado de la asignación en el archivo **Resultado.txt** (véase el texto resaltado). `RaptorXML xslt --xslt-version=2 --input="MappingMapToResult1.xslt" --output="Resultado.txt" %* "MappingMapToResult1.xslt"` Si no se definió el parámetro `--output`, 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.

5.6.1 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**.

5.6.2 Ejemplo: vista previa de resultados de una función

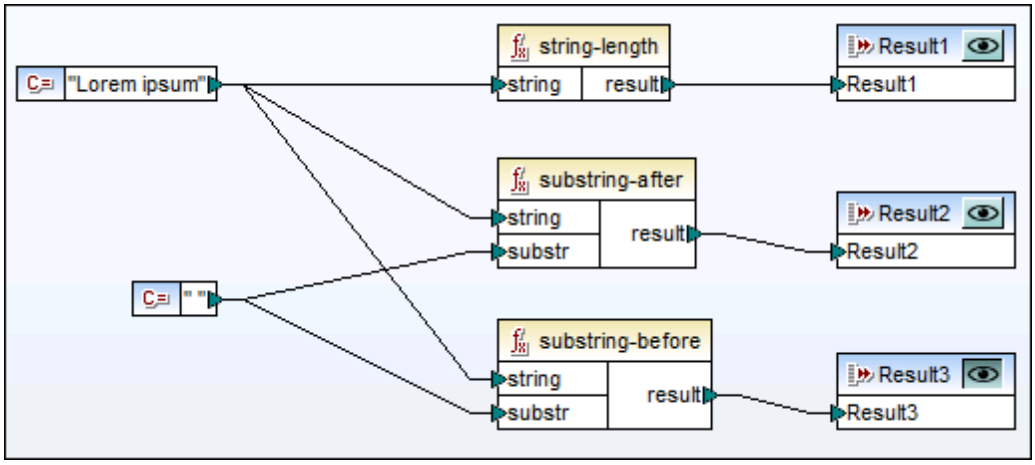
Este ejemplo sirve para explicar cómo consultar la vista previa de los resultados que devuelven funciones de MapForce con la ayuda de componente de salida simples. Antes de consultar este ejemplo es conveniente conocer el funcionamiento básico de las funciones en general y de las funciones de MapForce en particular. Puede encontrar más información en la sección Funciones.

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`	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`	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`	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.

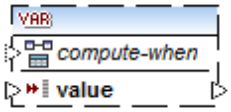
Para obtener la vista previa de resultados de una función:

- 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.

5.7 Usar variables

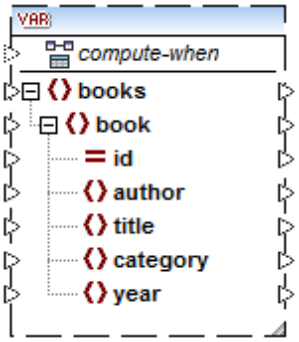
Las variables son un tipo especial de componente utilizado para almacenar un resultado de asignación intermedio y poder seguir procesándolo. Pueden ser necesarias cuando hace falta recordar temporalmente ciertos datos de la asignación y procesarlos de alguna manera (p. ej. filtrándolos o aplicándoles funciones) antes de copiarlos en el componente de destino.

Las variables pueden ser de tipo simple (p. ej. cadena, entero, valor booleano, etc) o de tipo complejo (estructuras jerárquicas).



Variable simple

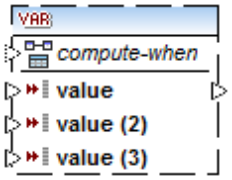
Puede crear una variable de tipo complejo suministrando un esquema XML que exprese la estructura de la variable. Si el esquema define elementos globalmente, podrá elegir cuál de ellos debe ser el nodo raíz de la estructura de la variable. Recuerde que una variable no tiene ningún archivo XML de instancia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



Variable compleja creada a partir de un esquema XML

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*).

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*). Sin embargo, esto no se puede hacer en las variables creadas a partir de tablas de BD.



Variable simple con entradas duplicadas

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). 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).

Hasta cierto punto las variables pueden compararse con los componentes intermedios de una asignación en cadena (*véase* Asignaciones en cadena). 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).
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  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.

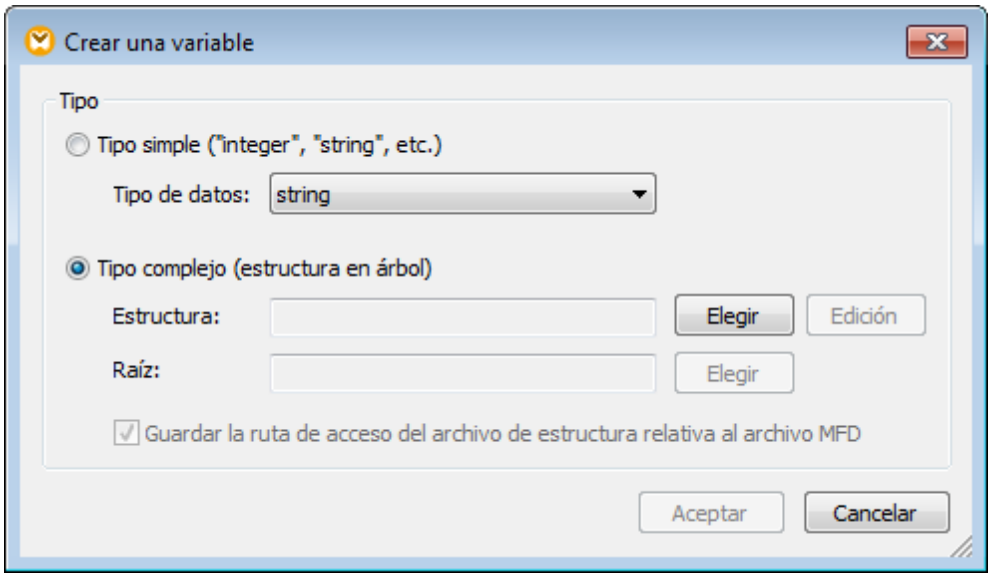
Nota: no se admite el uso de variables si el lenguaje de transformación elegido es XSLT 1.0.

5.7.1 Agregar variables

Hay varias maneras de agregar variables a una asignación de datos.

Con un comando de menú o de la barra de herramientas

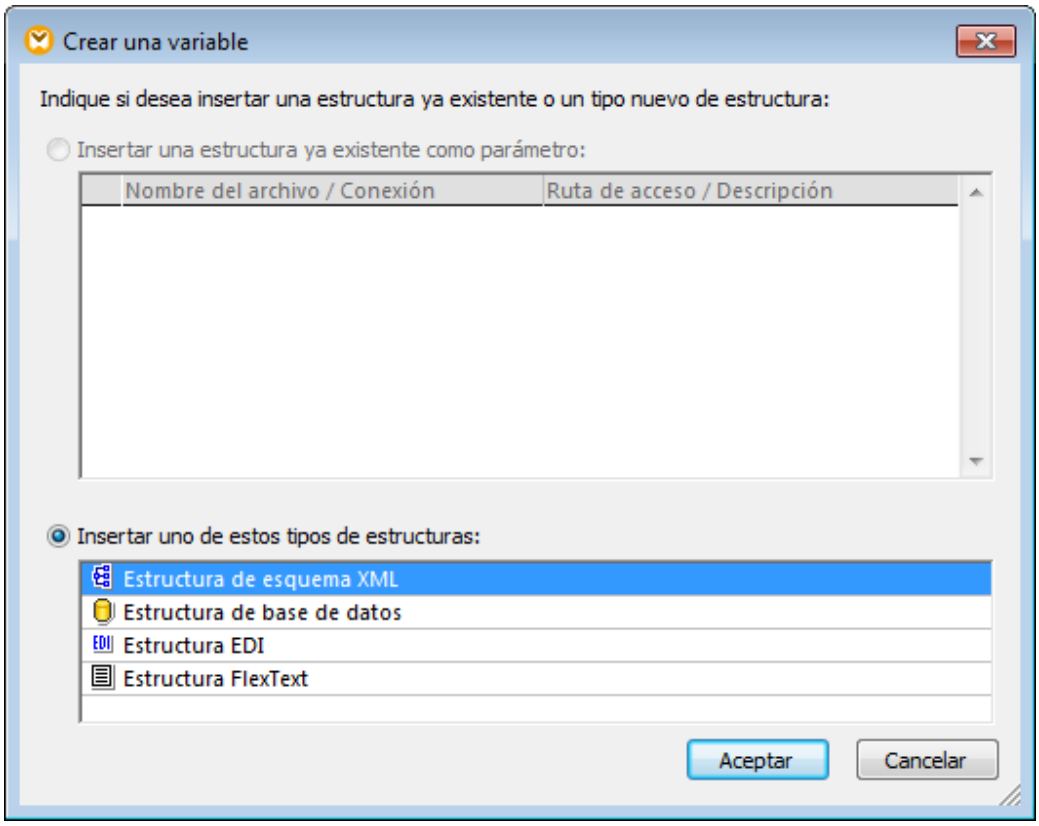
1. En el menú **Insertar** haga clic en el comando **Variable** (o haga clic en el botón **Variable**  de la barra de herramientas).



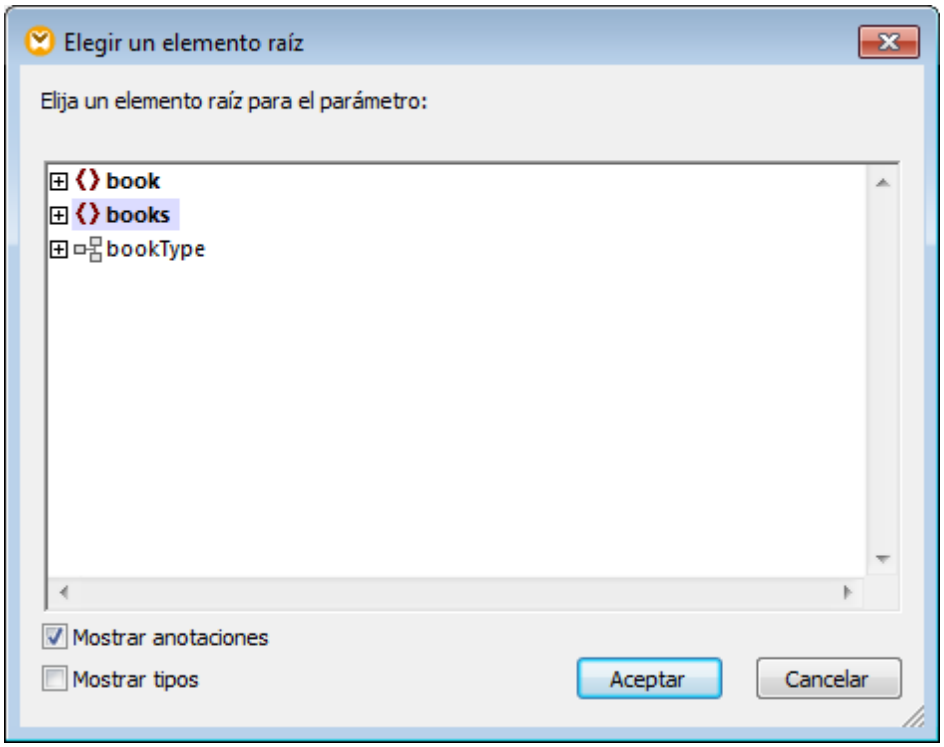
2. Seleccione el tipo de variable que desea insertar (de tipo simple o complejo).

   Si selecciona *Tipo complejo* debe seguir algunos pasos más:

3. Haga clic en **Elegir** para seleccionar la fuente que aportará la estructura de la variable (p. ej. un esquema XML).

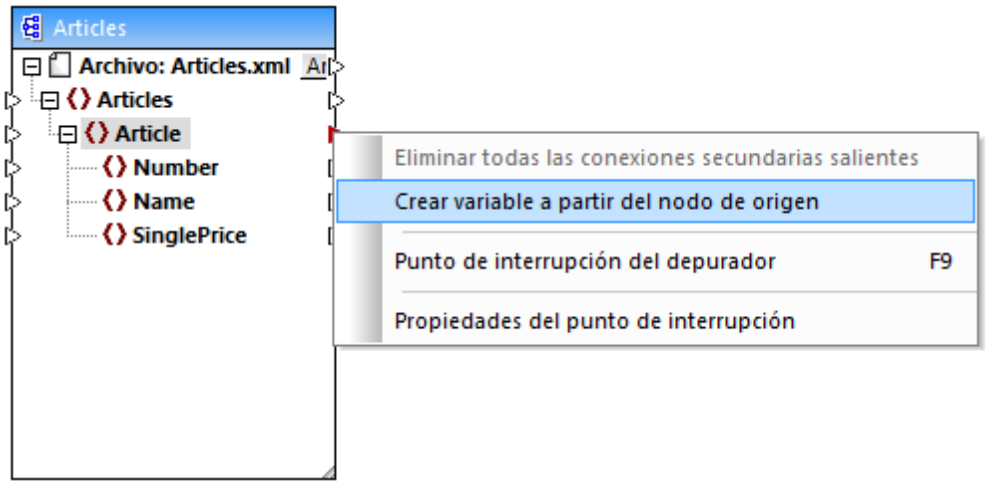


4. Cuando se solicite, especifique un elemento raíz de la estructura. En el caso de los esquemas XML, el elemento raíz puede ser cualquier elemento que esté definido globalmente. En el caso de las bases de datos, el elemento raíz puede ser cualquier tabla.

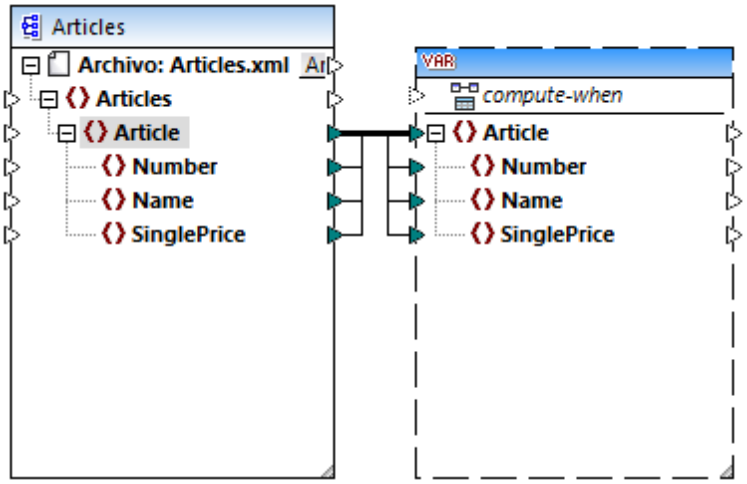


Con un menú contextual

- Haga clic con el botón derecho en el conector de salida de un componente (p. ej. "Article") y seleccione **Crear variable a partir del nodo de origen**.



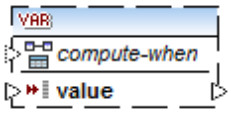
Esto crea una variable compleja que usa el mismo esquema de origen y conecta automáticamente todos los elementos con una conexión de copia total.



- Haga clic con el botón derecho en el conector de entrada de un componente de destino y seleccione **Crear variable para nodo de destino**. Esto crea una variable compleja que usa el mismo esquema que el componente de destino y conecta automáticamente todos los elementos con una conexión de copia total.
- 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**. Esto crea un componente complejo que usa el mismo esquema de origen y usa automáticamente como elemento raíz del componente intermedio el elemento vinculado a la entrada del filtro.

5.7.2 Cambiar el 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.



Elemento "compute-when"

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 complejas de funciones definidas por el usuario 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.

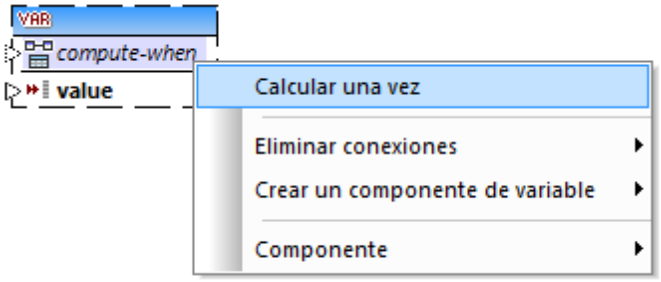
Si "Compute-when" está conectado

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`. 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*).

"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*).



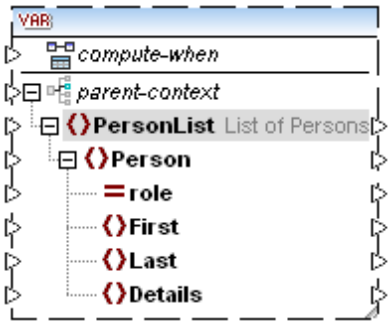
Cuando se cambia el ámbito de una variable por `compute-when=once`, el conector de entrada se elimina del elemento `compute-when` (porque dicha variable solamente se evaluará una vez).

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*).

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.



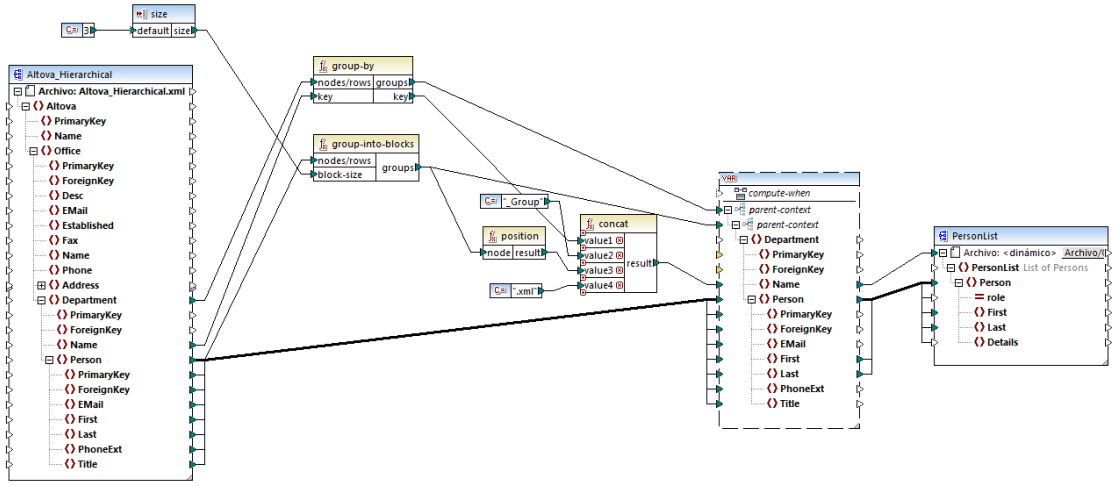
El contexto primario añade un nodo primario virtual a la jerarquía del componente, lo cual permite recorrer otro nodo más en el mismo componente de origen o en otro distinto.

5.7.3 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\MapForce2018\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*).

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`** 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`** 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*).

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`**, 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"

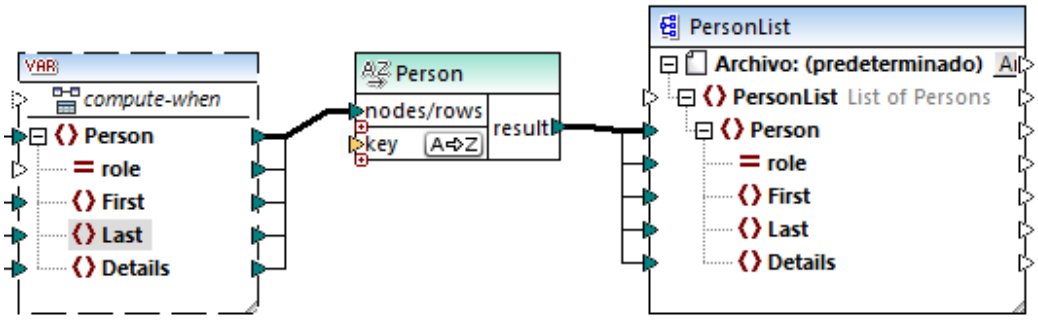
El resultado de la concatenación se almacena en el elemento `Name` de la variable y después se pasa como nombre de archivo dinámico al componente de destino. Por tanto, se crea un nombre de archivo nuevo por cada valor recibido. En este ejemplo, la variable calcula 8 grupos en total, así que se crean 8 archivos de salida cuando se ejecuta la asignación. Para más información consulte la sección Procesar varios archivos de entrada o salida simultáneamente.

5.8 Ordenar datos

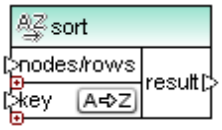
Para ordenar datos de entrada siguiendo un criterio de ordenación concreto basta con usar un componente de ordenación. El componente de ordenación es compatible con estos lenguajes de destino: XSLT2, XQuery y el motor de ejecución integrado.

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).



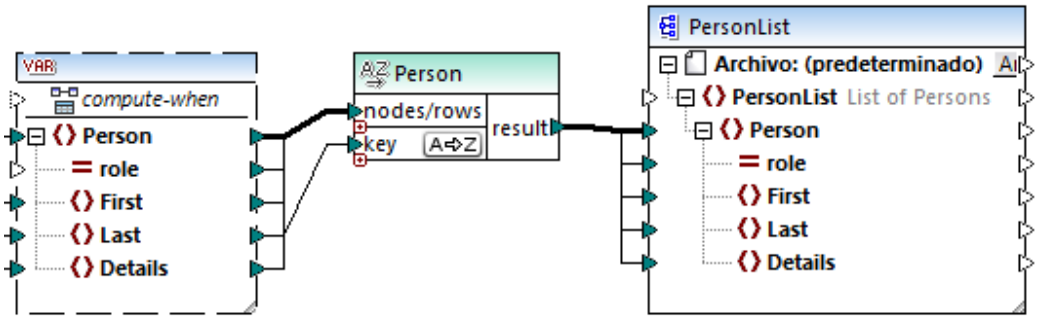
- En el menú **Insertar** seleccione el comando **Componente de ordenación: nodos/filas** (o haga clic en el botón **Insertar componente de ordenación** de la barra de herramientas). Esto inserta el componente de ordenación pero sin conectarlo.



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`.

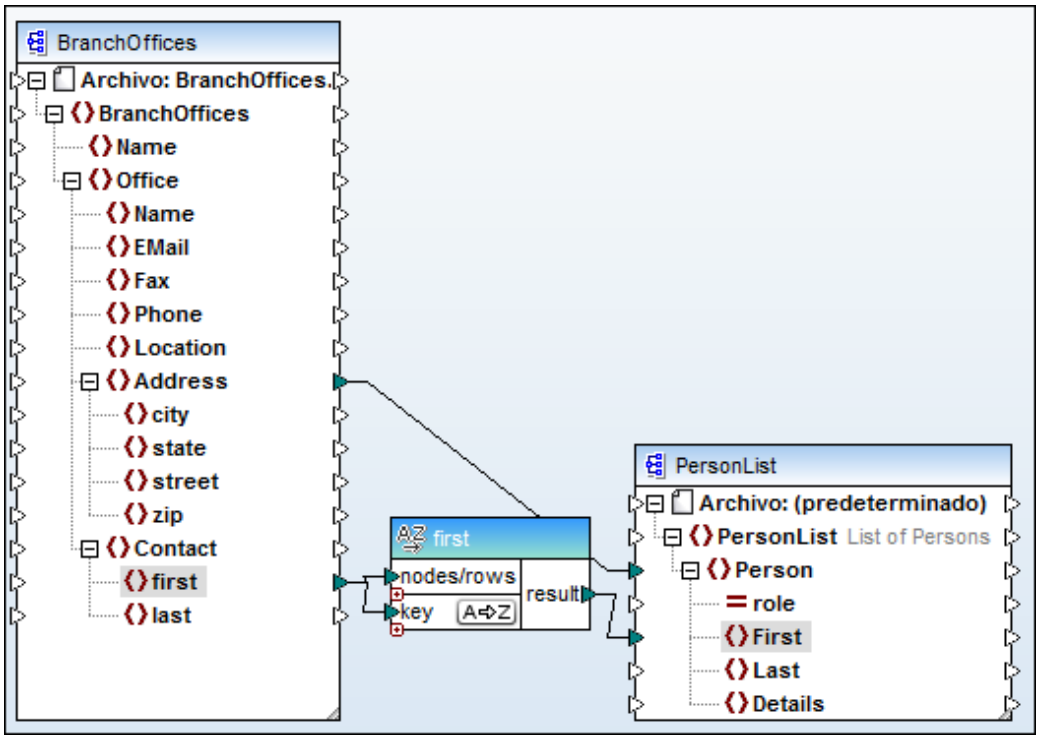


Para cambiar el orden:

- 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.

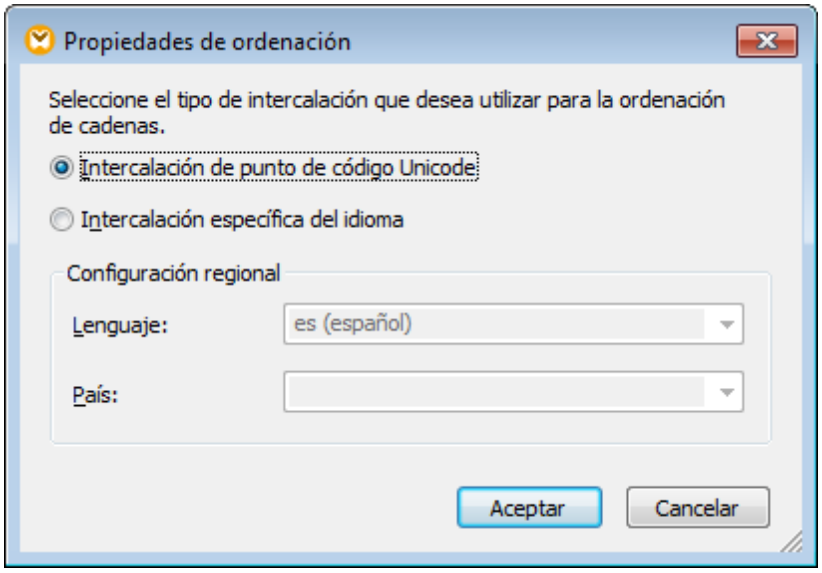
Para ordenar datos de entrada compuestos por elementos de tipo simple:

- 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*).

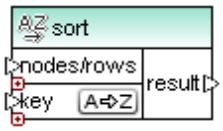


Intercalación de punto de código: esta opción predeterminada compara/ordena las cadenas basándose en valores de punto de código. Los valores de punto de código son enteros que se asignaron a caracteres abstractos del conjunto de caracteres universal adoptado por el consorcio Unicode. Esta opción permite 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.

5.8.1 Ordenar según varias claves

Cuando se añade un componente de ordenación a la asignación, MapForce crea por defecto un criterio de ordenación llamado `key`.

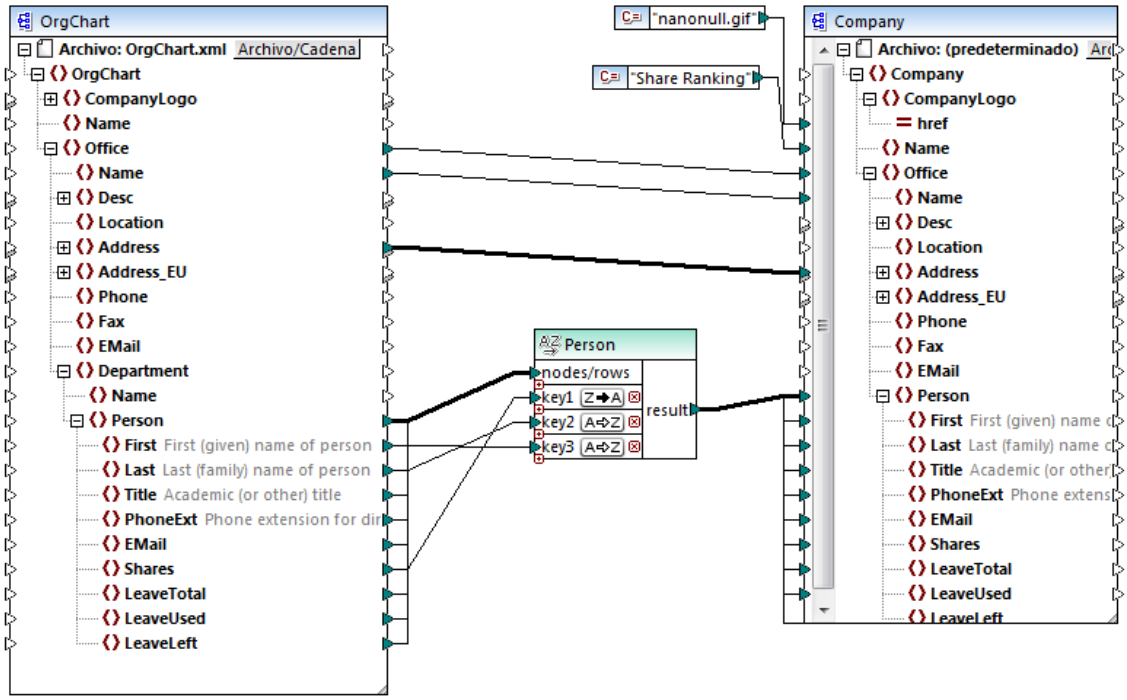


Componente de ordenación

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\MapForce2018\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:

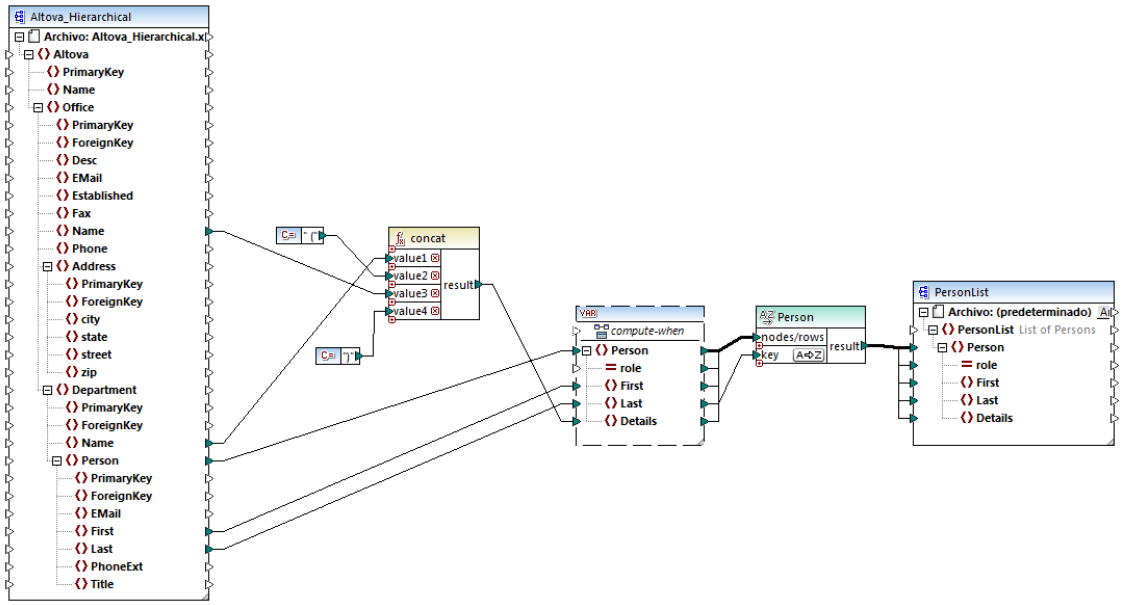
1. `Shares` (número de acciones en propiedad de cada persona)
2. `Last` (apellido)
3. `First` (nombre)

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).

5.8.2 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\MapForce2018\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 Trabajar con funciones*).
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*). 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 y Conexiones de copia total 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.

5.9 Filtros y condiciones

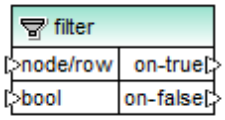
Cuando necesite filtrar datos o conseguir un valor de forma condicional, utilice uno de estos componentes:

- 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.

Devolver un valor de forma condicional

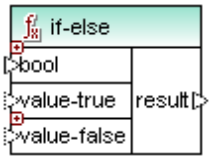
Si necesita obtener un solo valor (en lugar de un nodo o de una fila) de forma condicional, lo mejor es usar la **condición If-Else**. Recuerde que estas condiciones no son adecuadas para el filtrado de nodos o filas. A diferencia de los componentes **Filtro: nodos/filas**, las **condiciones If-Else** 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>
```

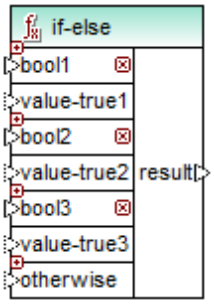
Una **condición If-Else** permite devolver por cada elemento de la lista el valor "alto" si la temperatura supera los 20 grados centígrados y el valor "bajo" si la temperatura es inferior a los 5 grados centígrados.

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.



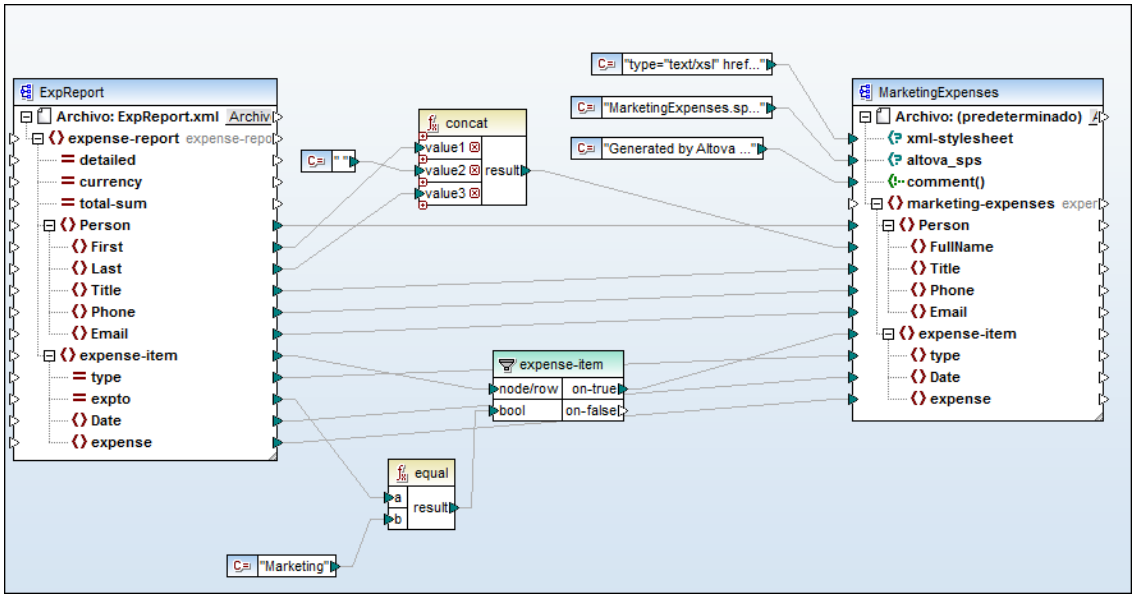
Las **condiciones If-Else** extendidas, se evalúan de arriba a abajo (se comprueba en primer lugar las condiciones situadas más arriba). Si desea devolver un valor aunque no se cumpla ninguna condición (es decir, aunque ninguna condición sea true), conecte el valor a **`otherwise`**.

Para ver un ejemplo con instrucciones paso a paso consulte el apartado Ejemplo: devolver un valor de forma condicional.

5.9.1 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\MapForce2018\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 Trabajar con funciones*). 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>
...
```

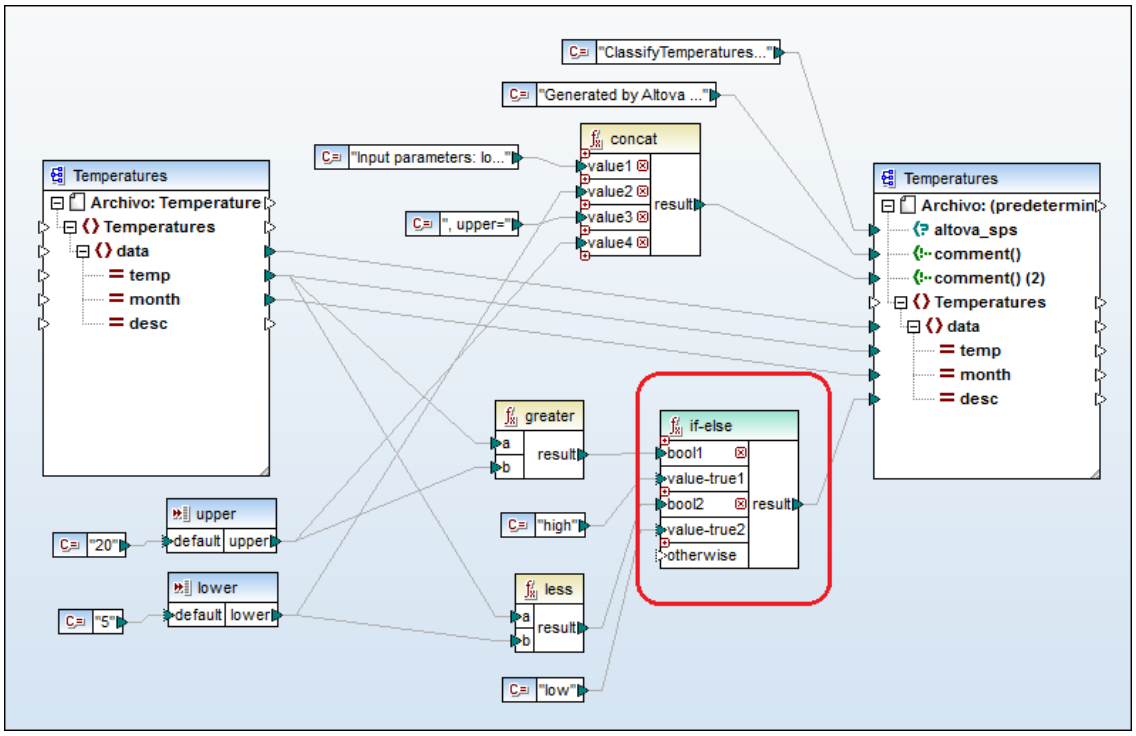
XML de salida después de ejecutarse la asignación

5.9.2 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*).

La asignación de este ejemplo es **<Documentos>\Altova\MapForce2018\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*). 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 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

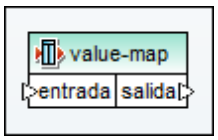
5.10 Usar asignaciones de valores

El componente Asignación de valores sirve para transformar un valor de entrada en un valor de salida distinto por medio de una tabla de consulta. Es un componente muy práctico a la hora de convertir diferentes tipos de enumeración. Solamente está compuesto por un elemento de entrada y otro de salida.

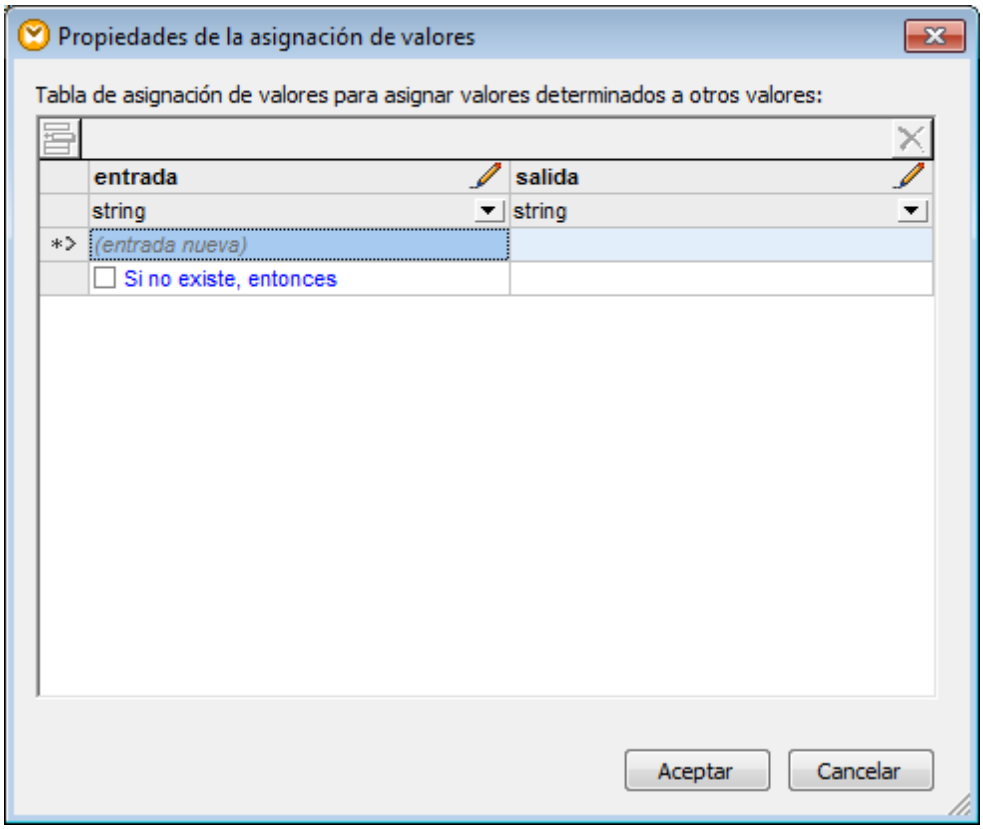
Nota: si desea recuperar/filtrar datos en base a determinados criterios, se recomienda el componente Filtro (*véase* *Filtros y condiciones*).

Para usar un componente Asignación de valores:

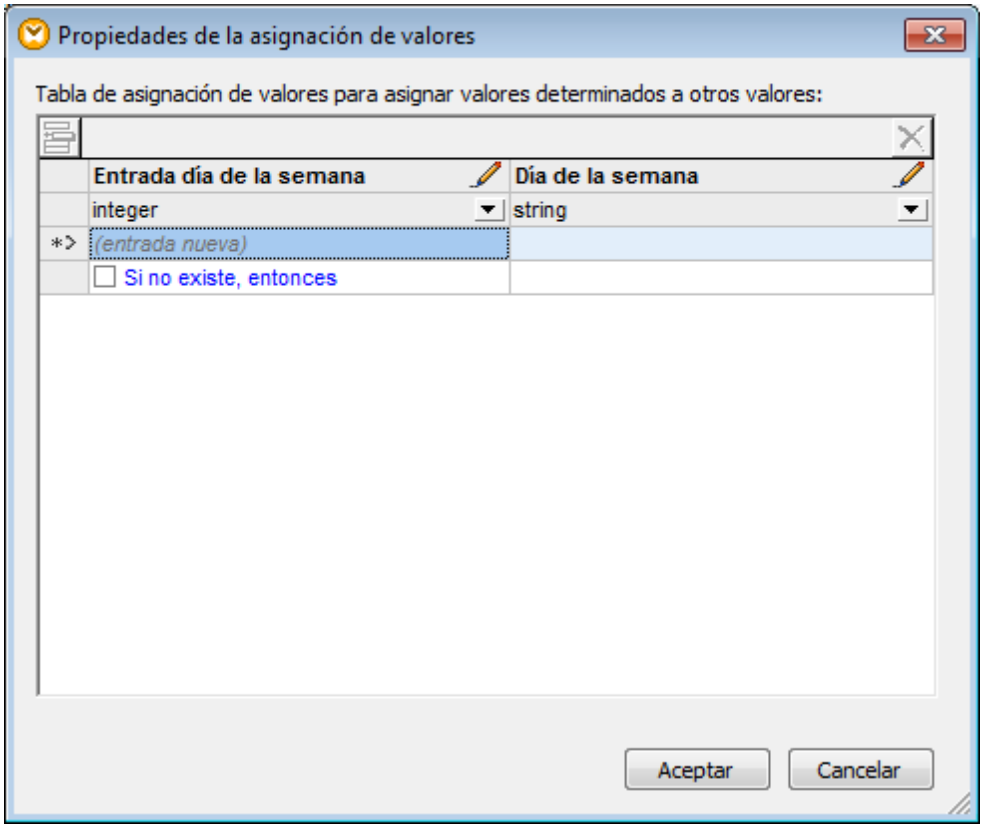
1. Seleccione el comando de menú **Insertar | Asignación de valores** o haga clic en el icono **Asignación de valores** de la barra de herramientas.



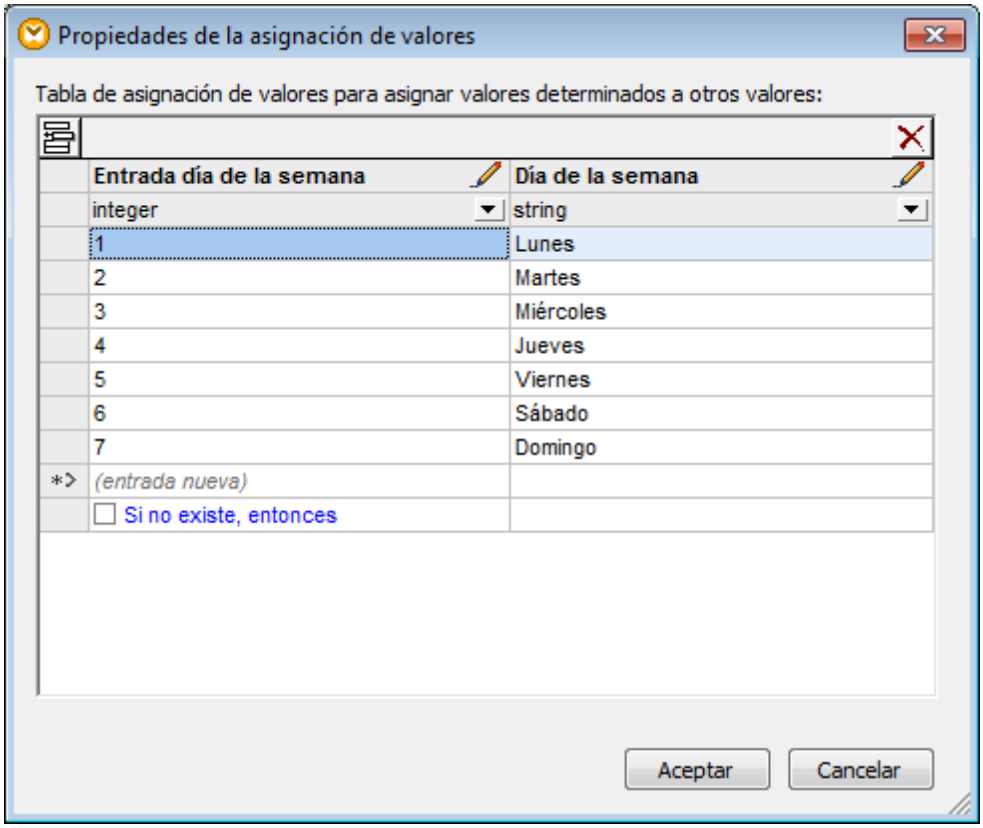
2. Haga doble clic en el componente **value-map** para abrir la tabla de asignación de valores.



3. Haga clic en los encabezados de columna e introduzca el texto `Entrada día de la semana` en el primer encabezado y `Día de la semana` en el segundo.



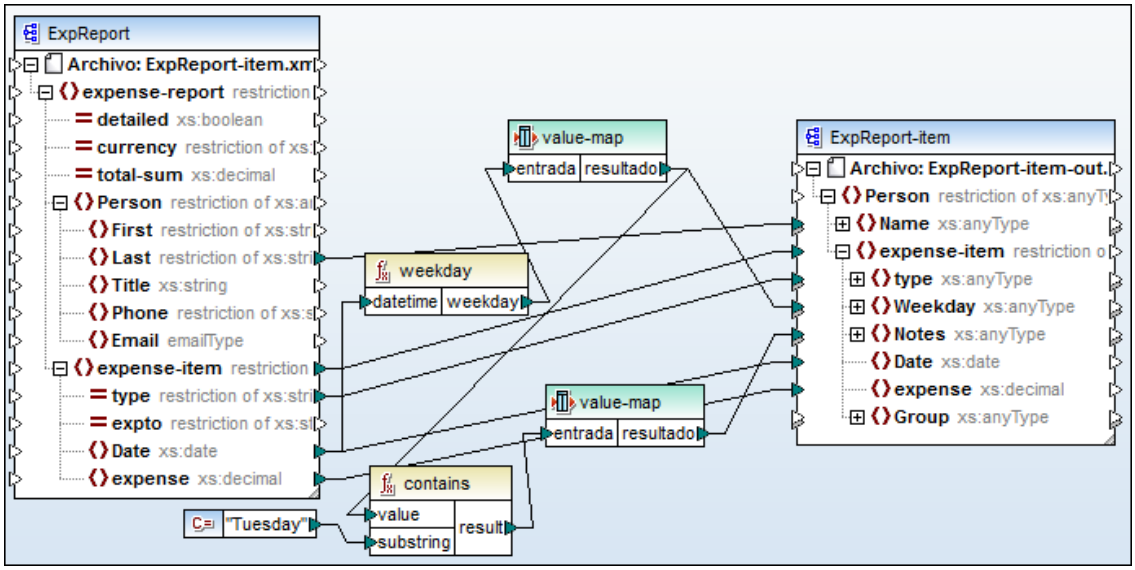
4. Ahora introduzca el valor de entrada que desea transformar en la columna `Entrada día de la semana`.
5. En la columna `Día de la semana` introduzca el valor de salida al que se debe transformar el valor de entrada.
6. Para introducir un par de valores nuevo introduzca texto en el campo de entrada *`(entrada nueva)`*.
7. Haga clic en el cuadro combinado de tipo de datos (situado justo debajo del encabezado de columna) para seleccionar los tipos de datos de entrada y salida (p. ej. `integer` y `string`).



Nota: para definir un valor de salida alternativo si los valores dados no existen en la entrada, marque la casilla `Si no existe, entonces` e introduzca el valor alternativo. Para aprender a pasar datos de origen sin modificarlos consulte el apartado Pasar datos sin modificarlos.

8. Para cambiar el nombre de las columnas (que también aparecen en el diseño de asignación) haga clic en los iconos de edición de las filas de encabezado. Esto le permitirá identificar con facilidad qué papel desempeña el componente en la asignación de datos.

El archivo **Expense-valmap.mfd** de la carpeta ...\MapForceExamples\Tutorial\ contiene un componente **Asignación de valores**.



¿Qué hace el diseño de asignación de datos **Expense-valmap.mfd**?
Extrae el día de la semana del elemento **Date** del origen de datos, convierte el valor numérico en texto y lo coloca en el elemento **Weekday** del componente de destino (es decir, Sunday, Monday, etc.).

- La función **weekday** extrae el número del día de la semana del elemento **Date** del archivo de origen **ExpReport**. El resultado de esta función son enteros comprendidos entre el 1 y el 7.
- La asignación de valores **value-map** transforma los enteros en días de la semana (es decir, Sunday, Monday, etc.) tal y como se explica al principio de este apartado.
- Si la salida contiene `Tuesday`, entonces se asigna el valor de salida correspondiente al elemento **Notes** del componente de destino.
- Si hacemos clic en el panel *Resultados* podemos ver el archivo XML de destino con los datos ya transformados.

```
3     <Name>Landis</Name>
4     <expense-item>
5       <type>Meal</type>
6       <Weekday>Tuesday</Weekday>
7       <Notes>-- Prepare financial reports -- !</Notes>
8       <Date>2003-01-01</Date>
9       <expense>122.11</expense>
10    </expense-item>
11    <expense-item>
12      <type>Lodging</type>
13      <Weekday>Monday</Weekday>
14      <Notes> --</Notes>
15      <Date>2003-01-14</Date>
16      <expense>122.12</expense>
17    </expense-item>
```

Nota: si pasamos el puntero por encima del componente value-map, aparece información rápida con los valores que están definidos.

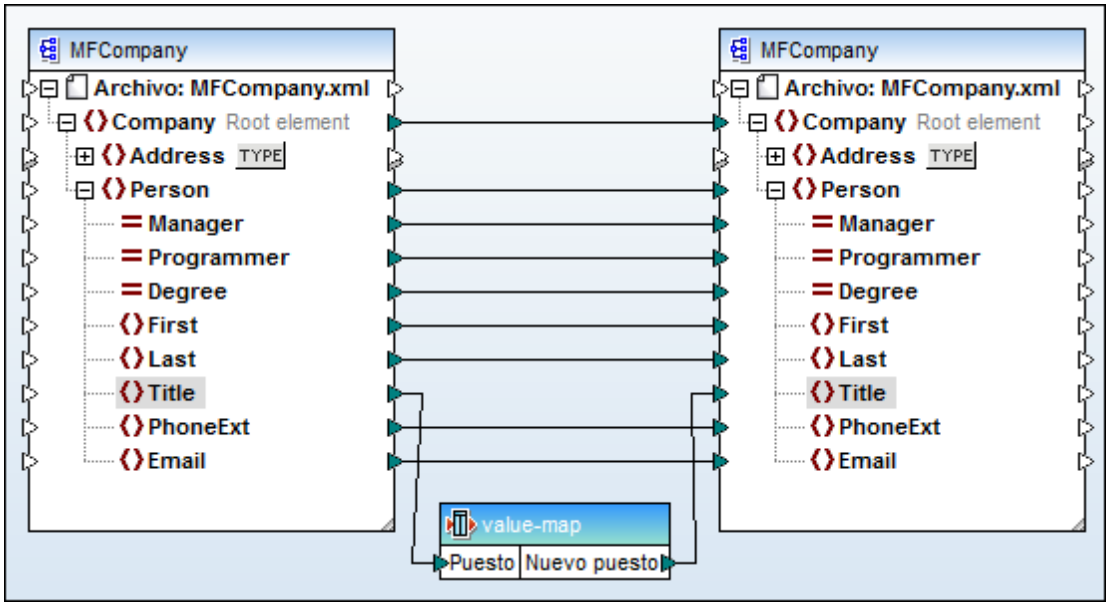
El resultado de varias funciones de cadena y funciones lógica solamente puede ser el valor

booleano **true** o **false**. Por tanto, el valor que se desea comprobar debe introducirse en el campo entrada de la tabla de la asignación de valores (p. ej. true).

5.10.1 Pasar datos sin modificarlos

En algunos casos es necesario transformar datos concretos de un nodo, mientras que el resto de datos del nodo deben pasarse al nodo de destino sin modificaciones.

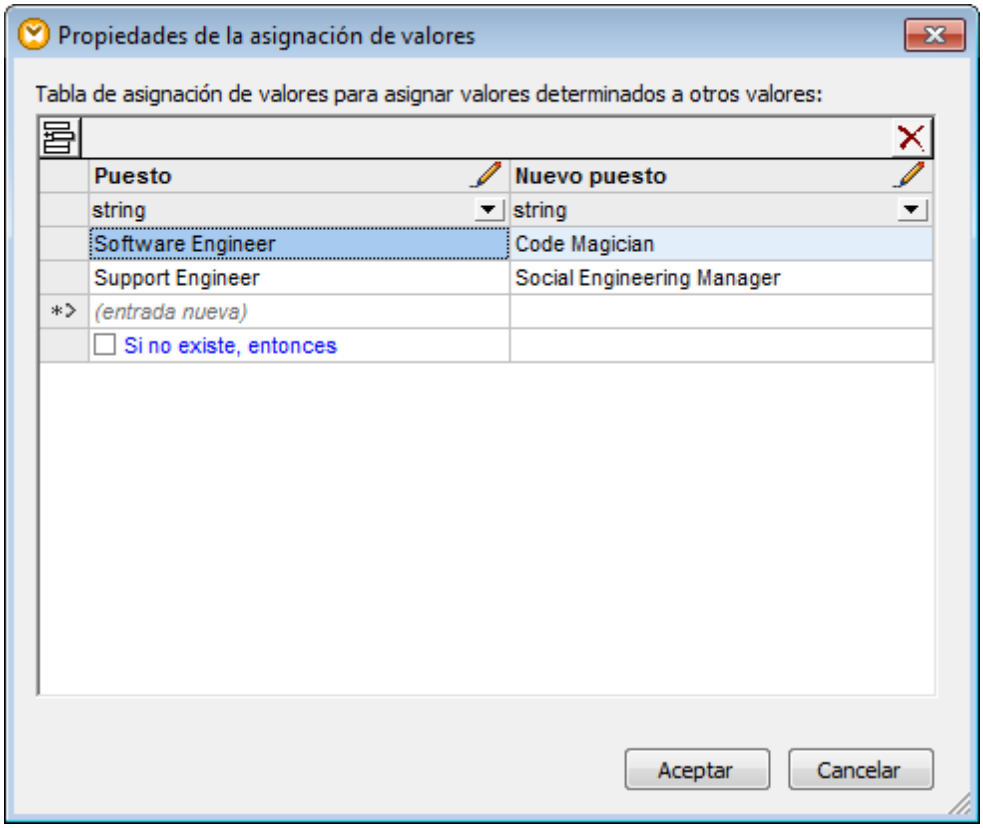
Por ejemplo, imaginemos que en una filial de una compañía cambian los nombres de los puestos de los empleados. En nuestro ejemplo cambian dos nombres de puestos pero el resto de puestos no cambian.



La asignación más obvia sería la que aparece en la imagen anterior, que utiliza una asignación de valores para transformar el nombre de los puestos.
Si abrimos el panel *Resultados* podremos ver el resultado de la asignación:

```
33  <Person>
34    <First>Fred</First>
35    <Last>Landis</Last>
36    <PhoneExt>951</PhoneExt>
37    <Email>f.landis@nanonull.com</Email>
38  </Person>
39  <Person>
40    <First>Michelle</First>
41    <Last>Butler</Last>
42    <Title>Code Magician</Title>
43    <PhoneExt>654</PhoneExt>
44    <Email>m.landis@nanonull.com</Email>
45  </Person>
```

En el caso de los empleados que no tengan ninguno de los dos tipos que muestra la asignación de valores, el elemento **Title** se eliminará en el documento de salida.

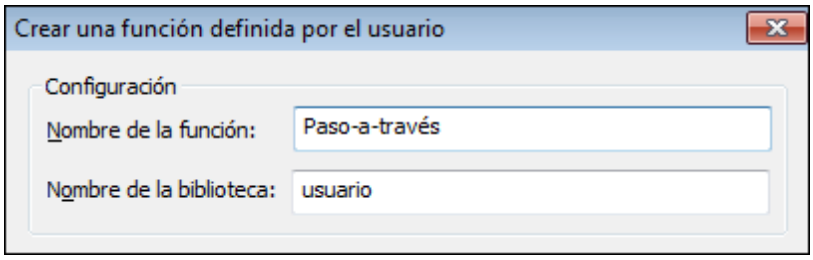


Otra manera de hacer esto sería marcar la casilla *Si no existe, entonces* e introducir un término de sustitución. El nodo **Title** volvería a aparecer en el documento de salida, pero ahora incluiría el mismo `Nuevo puesto` que los demás empleados de la compañía.

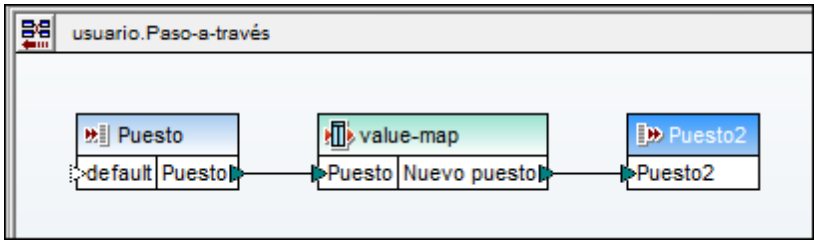
Solución:

Cree una función definida por el usuario que contenga un componente de asignación de valores y utilice la función **substitute-missing** para suministrar los datos iniciales para los nodos vacíos.

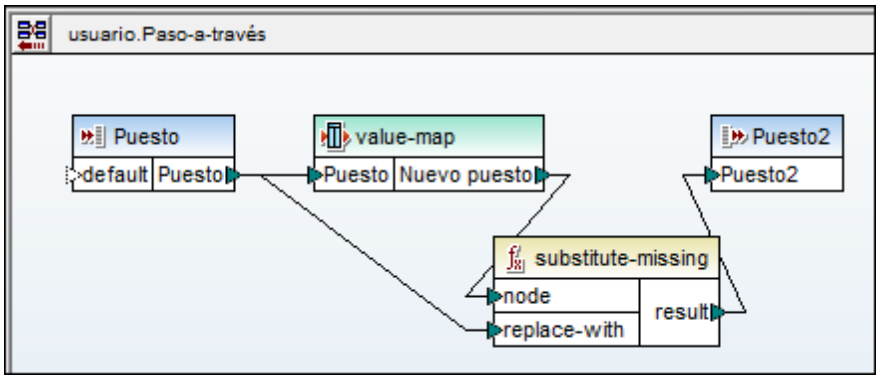
1. Haga clic en la asignación de valores **value-map** y seleccione **Función | Crear una función definida por el usuario a partir de la selección**.



2. Introduzca un nombre para esta función (p. ej. Paso-a-través) y haga clic en **Aceptar**.



3. Inserte una función **substitute-missing** desde la sección **core | node function** del panel Bibliotecas y cree las conexiones que aparecen en esta imagen:



4. Ahora puede abrir el panel *Resultados* para ver el resultado.

Resultado de la asignación:

- Las dos designaciones Title de la asignación de valores value-map se transformaron en Nuevo puesto.
- Los demás nodos Title del archivo de origen conservan sus datos iniciales en el archivo de destino.

```
38  <Person>
39    <First>Fred</First>
40    <Last>Landis</Last>
41    <Title>Program Manager</Title>
42    <PhoneExt>951</PhoneExt>
43    <Email>f.landis@nanonull.com</Email>
44  </Person>
45  <Person>
46    <First>Michelle</First>
47    <Last>Butler</Last>
48    <Title>Code Magician</Title>
49    <PhoneExt>654</PhoneExt>
50    <Email>m.landis@nanonull.com</Email>
51  </Person>
```

¿Por qué ocurre esto?
La asignación de valores value-map evalúa los datos de entrada de la siguiente manera:

- Si los datos entrantes **coinciden con una** de las entradas de la primera columna, los datos se transforman y se pasan al parámetro `node` de la función **substitute-missing** y después a `Puesto2`.

- Si los datos entrantes **no coinciden con ninguna** entrada de la columna izquierda, no se pasa ningún dato de la asignación de valores al parámetro `node` (es decir, es un nodo vacío).

  Cuando esto ocurre, la función **substitute-missing** recupera el nodo y los datos iniciales del nodo `Title` y los pasa a través del parámetro **replace-with** hasta `Puesto2`.

5.10.2 Propiedades de las asignaciones de valores

Acciones:

 Este icono **inserta** una fila nueva antes de la fila activa.

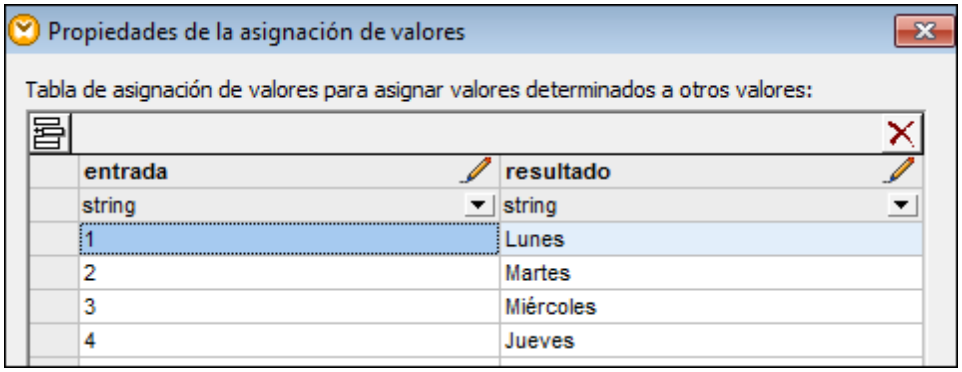
 Este icono **elimina** la fila activa.

 Este icono sirve para **editar** el encabezado de columna.

También puede cambiar el orden de las filas arrastrándolas y colocándolas en una nueva posición.

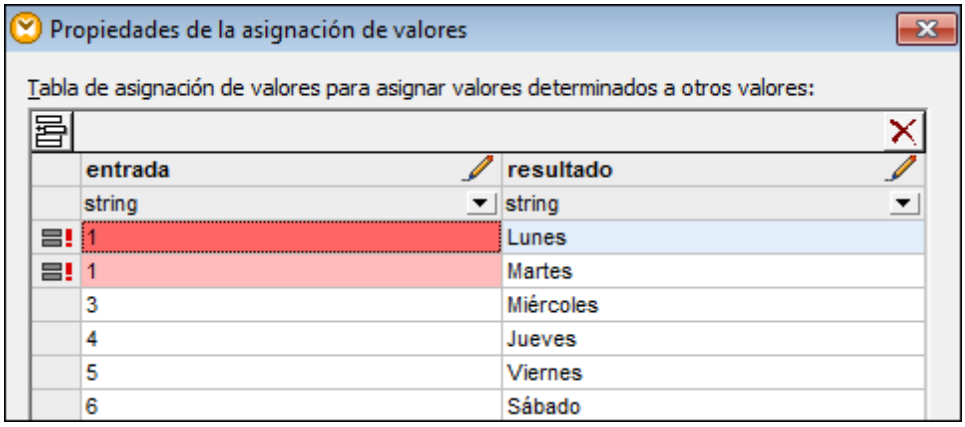
Cambiar el encabezado de columna:

Haga doble clic en el encabezado de columna (o clic en el icono ) para editar el nombre de la columna. Los nombres de las columnas aparecen en el diseño de asignación y así podrá identificar con facilidad qué papel desempeña el componente.



Usar valores de entrada únicos:

Los valores que introduzca en la columna de entrada deben ser valores únicos. Si introduce dos valores idénticos, ambos se resaltan en rojo para que los corrija.

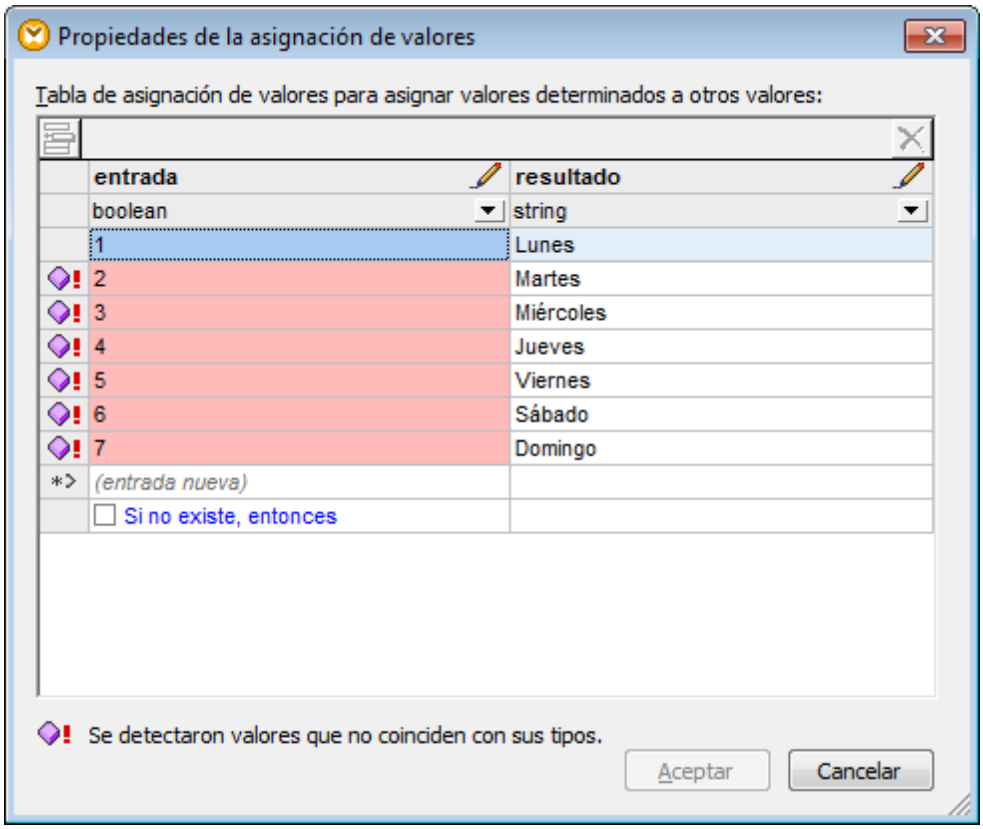


Una vez corregidos los valores, el botón **Aceptar** del cuadro de diálogo se habilita de nuevo.

Tipos de datos de entrada y salida

Los tipos de datos de entrada y salida se verifican automáticamente cuando se selecciona una entrada en el cuadro combinado. Si hay un error de coincidencia, se resaltan en rojo los campos pertinentes y se deshabilita el botón **Aceptar** del cuadro de diálogo.

En la imagen siguiente, por ejemplo, se seleccionaron los tipos de datos `boolean` y `string`.



5.11 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 solo 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** y **static-node-name** 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 solo 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*

* Solo compatible con MapForce Professional or Enterprise Edition.

Puede trabajar con nombres de nodo dinámicos si selecciona estos lenguajes de transformación: motor integrado BUILT-IN*, XSLT2, XQuery*, C#*, C++*, Java*.

* Solo compatible con MapForce Professional o Enterprise Edition.

Para más información consulte el apartado Obtener acceso al nombre de los nodos o el ejemplo del apartado Ejemplo: asignar nombres de elemento a valores de atributo.

5.11.1 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 solo 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\MapForce2018\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\MapForce2018\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.

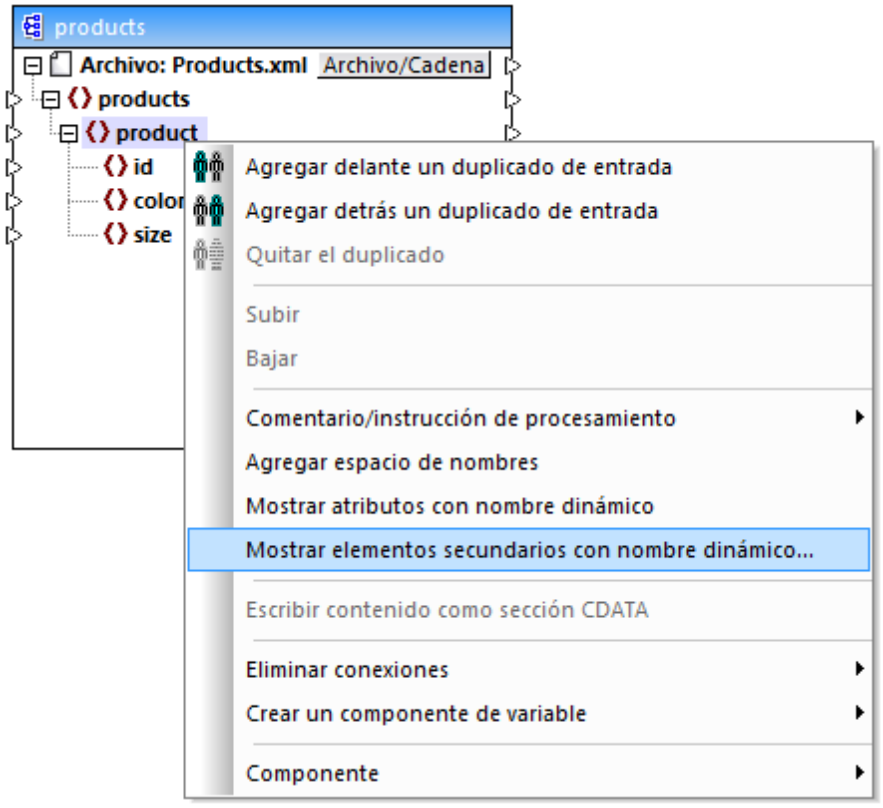


Fig. 1: habilitar nombres de nodo dinámicos (para elementos secundarios)

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).

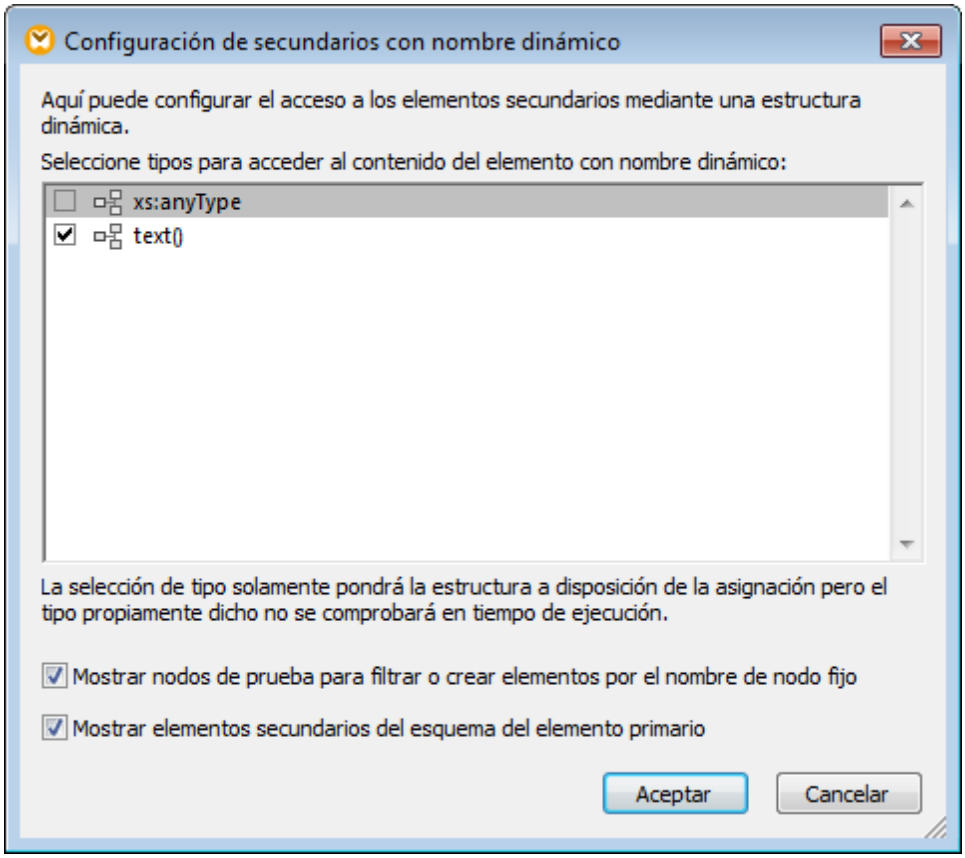


Fig. 2: cuadro de diálogo "Configuración de secundarios con nombre dinámico"

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.

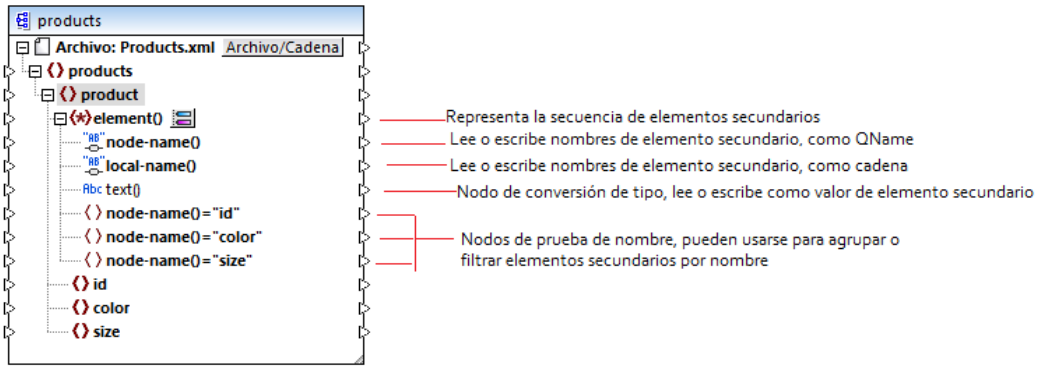


Fig. 3: nombres de nodo dinámicos habilitados (para elementos secundarios)

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).

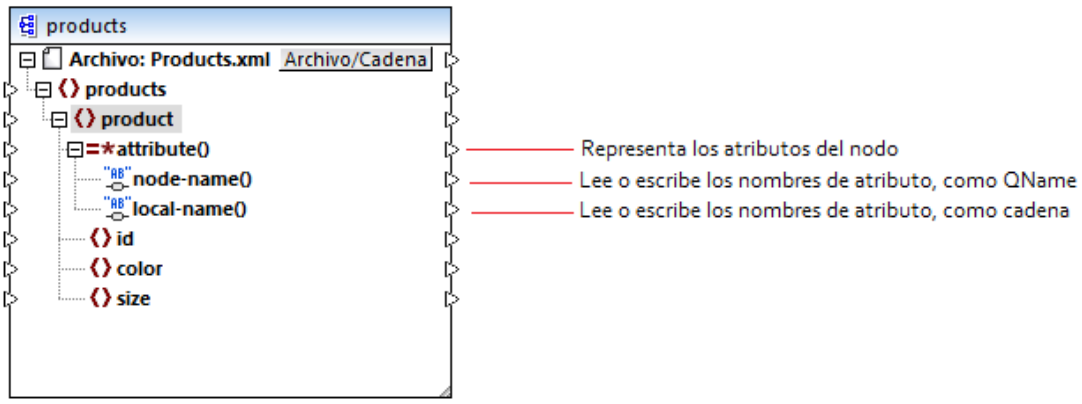


Fig. 4: nombres de nodo dinámicos habilitados (para atributos)

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:

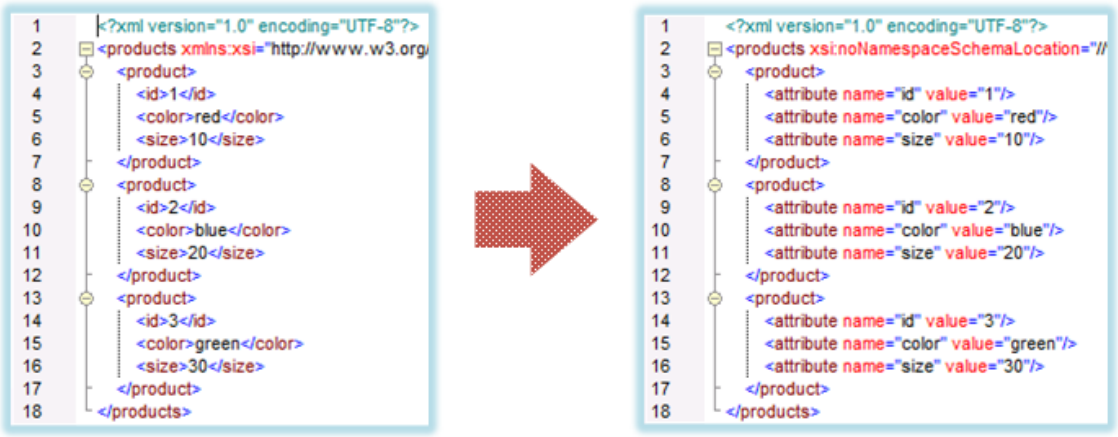


Fig. 6: asignación entre nombres de elemento XML y valores de atributo

Para conseguir este objetivo debemos diseñar esta asignación de datos:

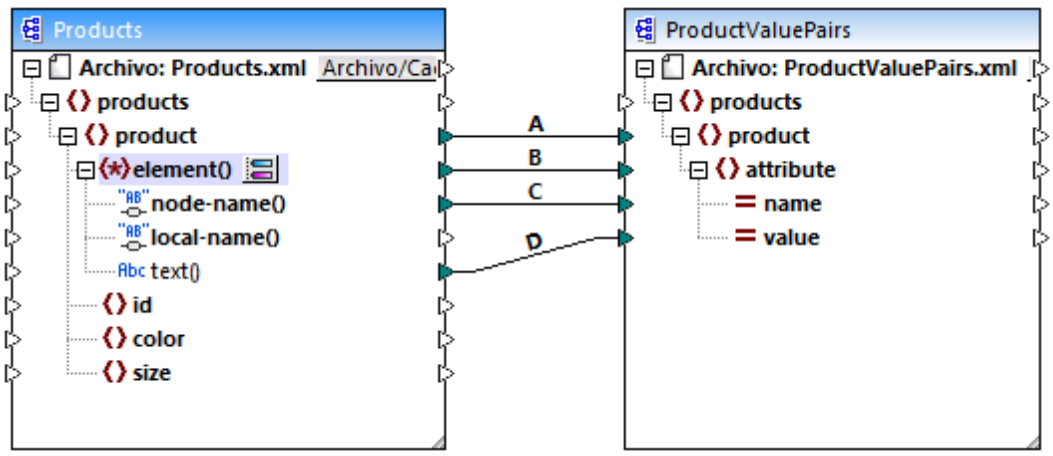


Fig. 7: asignar nombres de elemento XML a valores de atributo

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.

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()` solo 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:

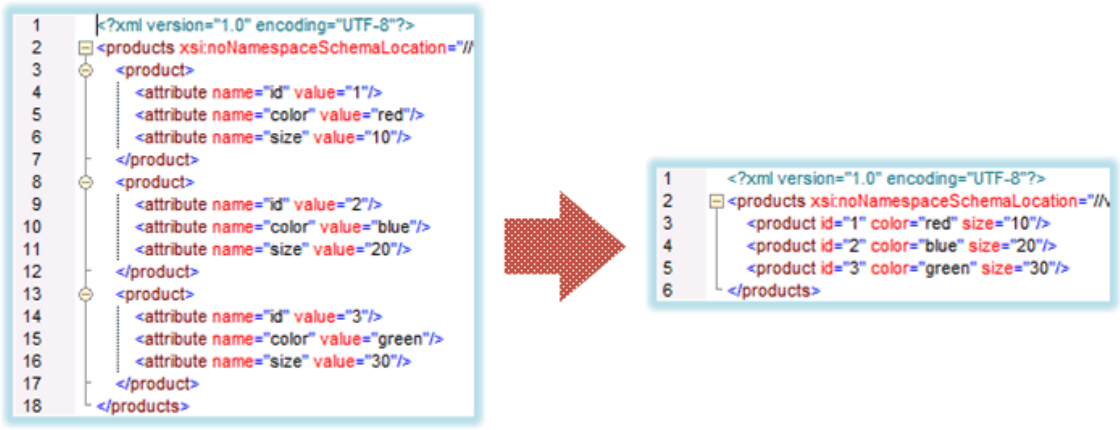


Fig. 8: asignar valores a nombres de atributo

Para conseguir este objetivo debemos diseñar esta asignación de datos:

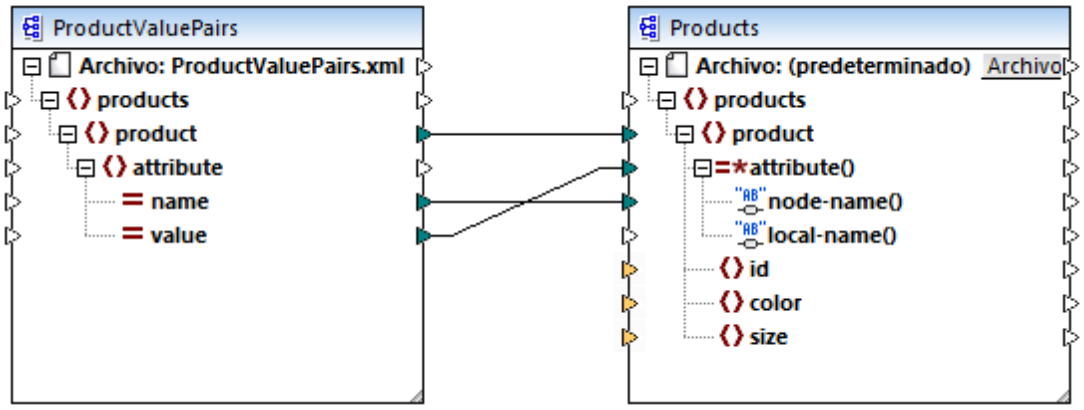


Fig. 9: asignar valores de atributo a nombres de atributo

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\MapForce2018\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 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*).

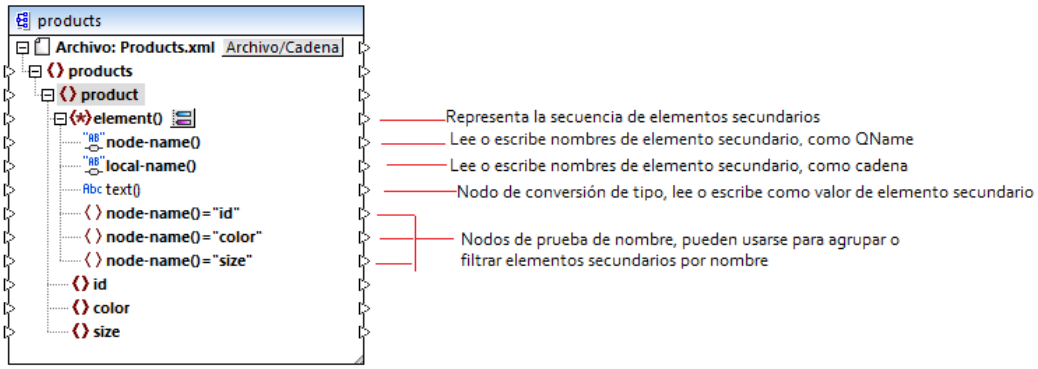
En general los nodos de prueba de nombre funcionan casi igual que los nodos de elemento normales para leer y escribir valores y subestructuras. Sin embargo, como la semántica de la asignación es distinta cuando está habilitado el acceso dinámico, existen algunas restricciones. Por ejemplo, no se puede 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()`.

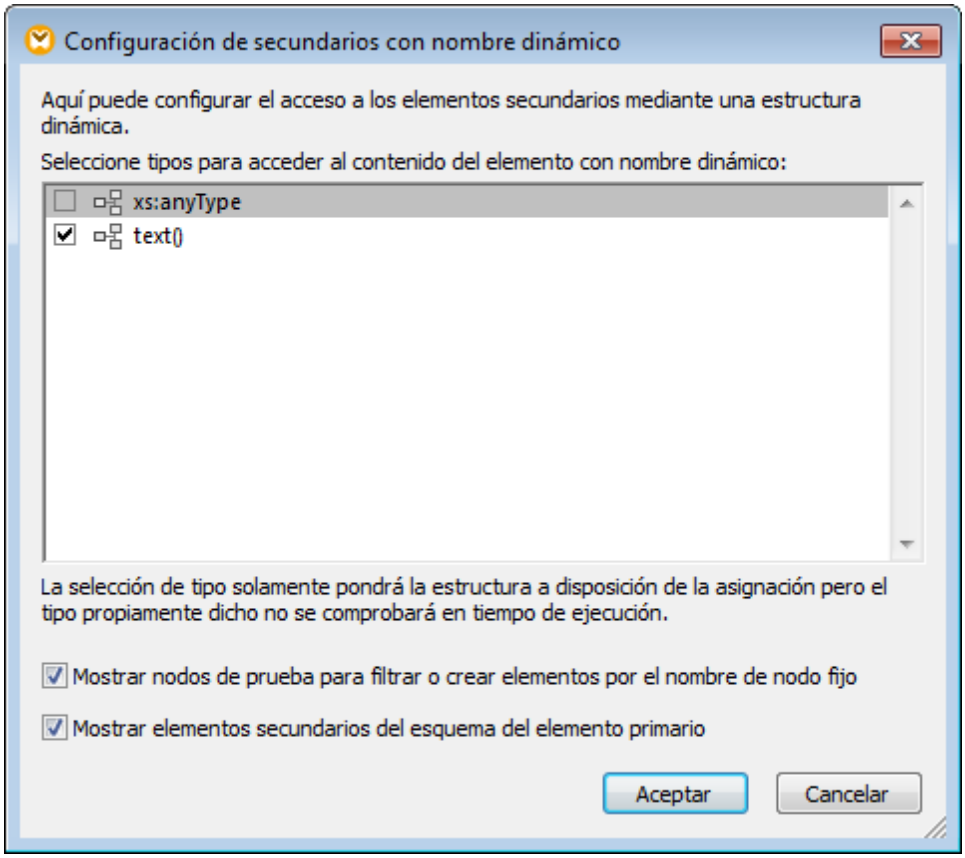
Si es necesario, puede ocultar los nodos de prueba de nombre en el componente. Esto se hace haciendo clic con el botón **Cambiar selección** del nodo `element()` y desactivando la casilla *Mostrar nodos de prueba....* en el cuadro de diálogo "Configuración de secundarios con nombre dinámico" (figura nº2).

5.11.2 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. 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:

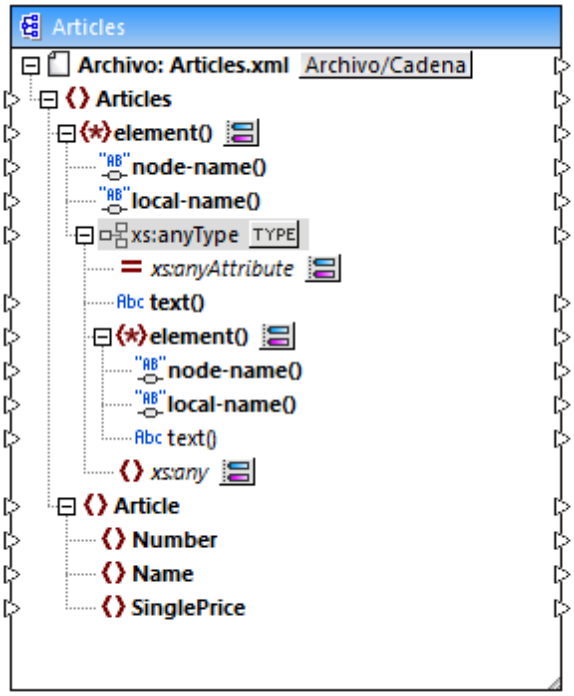
a) Asignar al nodo de conversión de tipo el tipo complejo que debe corresponderle.
b) Agregar un filtro para leer de la instancia solamente el tipo complejo que debe corresponder.

Para más información sobre los filtros consulte Filtros y condiciones.

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 esta técnica no es necesaria porque la asignación solo 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 \MapForce2018\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*).
- 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*). Si usa esta opción, asegúrese de que el esquema permite que el atributo o elemento exista de verdad.

5.11.3 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\MapForce2018\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 solo los valores) y estos nombres se podrán usar como valores. A continuación explicamos cómo diseñar esta asignación.

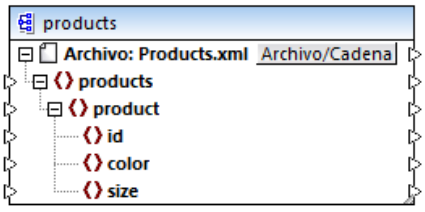
Paso nº1: agregar el componente XML de origen a la asignación

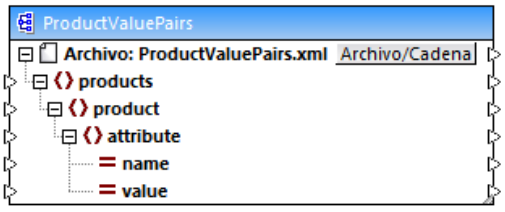
- En el menú **Insertar** haga clic en el comando **Archivo o esquema XML** y navegue hasta el archivo **<Documentos>\Altova\MapForce2018\MapForceExamples\Tutorial\Products.xml**. Este archivo XML apunta al esquema **Products.xsd** que está situado en la misma carpeta.

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\MapForce2018\MapForceExamples\Tutorial\ProductValuePairs.xsd**. Cuando la aplicación solicite un archivo de instancia, haga clic en **Omitir**. Cuando solicite un elemento raíz, seleccione `products`.

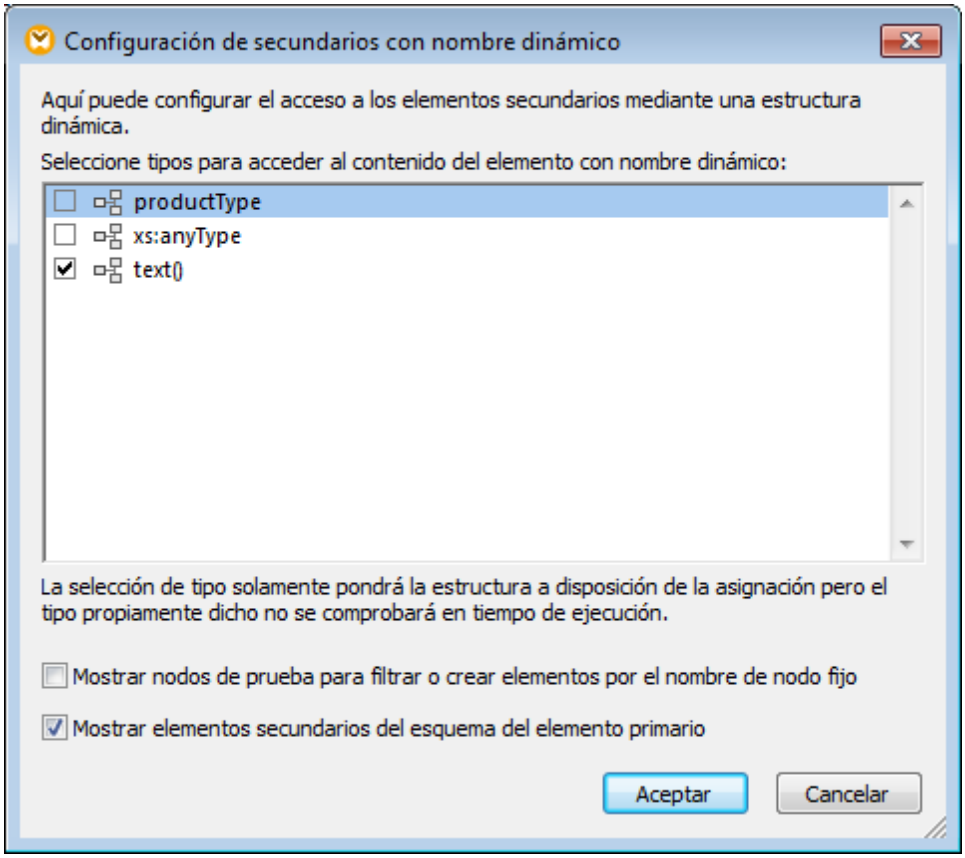
Llegados a este punto el diseño tendrá este aspecto:



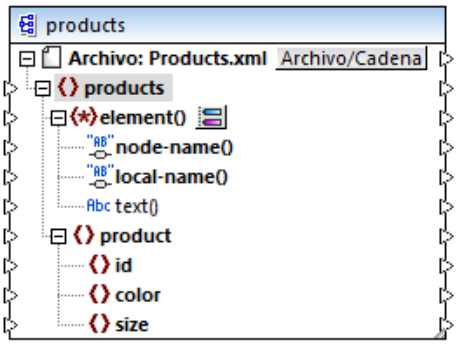


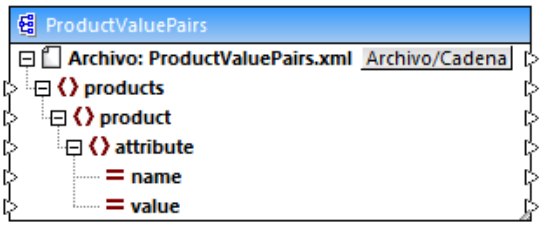
Paso nº3: habilitar el acceso dinámico a los nodos secundarios

1. Haga clic con el botón derecho en el nodo `products` 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.



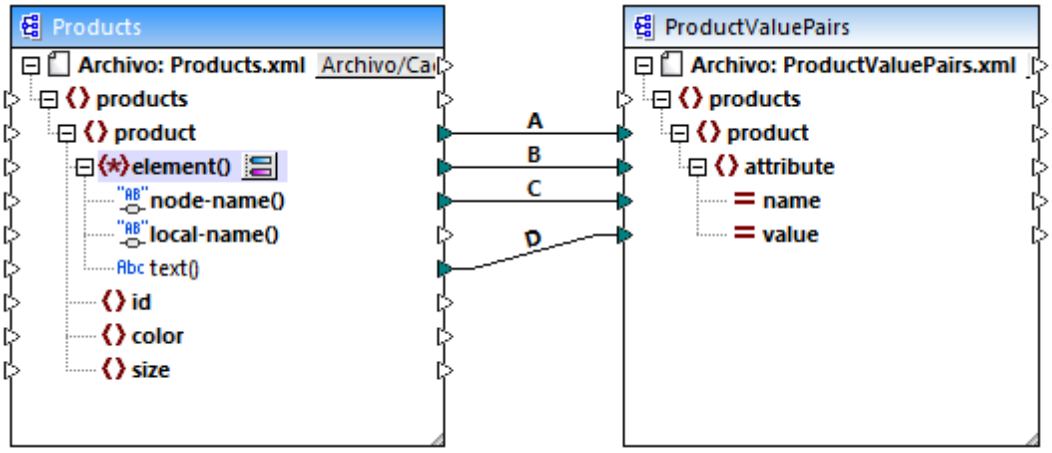
Observe que esto añade un nodo llamado `text()` al componente de origen. Este nodo se encargará de suministrar el contenido de los elementos secundarios a la asignación (en este caso, el valor de "id", "color" y "size").





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

5.12 Reglas y estrategias de asignación de datos

Por lo general MapForce crea asignaciones de datos de forma intuitiva pero en algunas ocasiones puede que los resultados tengan demasiados elementos o muy pocos. En este tema ofrecemos consejos para evitar este tipo de problemas.

Regla general

Por lo general, cada conexión entre un elemento de origen y uno de destino significa que, por cada elemento de origen, se creará un elemento de destino. Si el nodo de origen incluye contenido simple (p. ej. una cadena o un entero) y el nodo de destino acepta contenido simple, MapForce copia el contenido en el nodo de destino y, si hace falta, convierte el tipo de datos.

Esta es la regla general para todas las conexiones excepto en estos casos:

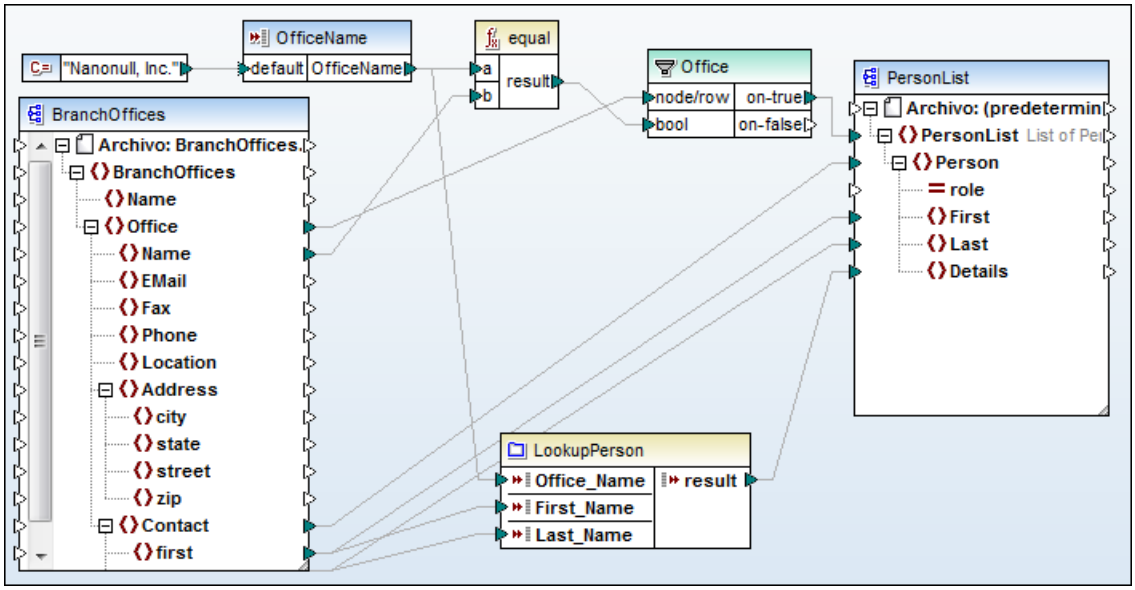
- El elemento raíz XML de destino se crea una sola vez. Si conecta una secuencia con el elemento raíz XML de destino, solo se repetirá el contenido del elemento pero no el elemento raíz propiamente dicho y puede que el resultado no sea válido según el esquema. Si también se conectan los atributos del elemento raíz, la serialización XML fallará en tiempo de ejecución, así que debe evitarse conectar una secuencia con el elemento raíz. Si lo que desea es crear varios archivos de salida, conecte la secuencia con el nodo `Archivo`, por medio de algún tipo de función que genere nombres de archivo.
- Algunos nodos aceptan un solo valor y no una secuencia (p. ej. los atributos XML y los componentes de destino de las funciones definidas por el usuario.

Los elementos "context" y "current"

MapForce muestra la estructura de un archivo de esquema en el componente como una jerarquía con elementos que se pueden asignar. Cada uno de estos nodos puede tener varias instancias (o ninguna) en el archivo de instancia o de la base de datos.

Ejemplo: en el componente de origen de la asignación **PersonListByBranchOffice.mfd** solo hay un nodo **first** (dentro de **Contact**). En el archivo de instancia **BranchOffices.xml**, hay varios nodos **first** y **Contact** con contenido distinto (dentro de los nodos primarios **Office**).

Dependiendo de cuál sea el nodo de **contexto** actual (del nodo de **destino**), se seleccionarán unos nodos de origen u otros y sus datos se copiarán, por medio del conector, en el componente/elemento de destino.



PersonListByBranchOffice.mfd

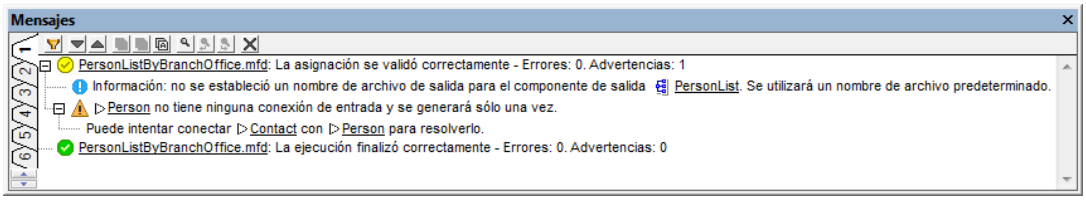
Este contexto viene definido por el **nodo de destino actual** y por las conexiones con sus antecesores:

- Al principio el contexto solo contiene los componentes de origen, pero ningún nodo concreto. Cuando evalúa la asignación MapForce procesa primero el nodo **raíz de destino** (PersonList) y después sigue recorriendo la jerarquía.
- El conector que está unido al nodo de **destino** se sigue hasta todos los elementos de origen a los que está conectado (directa o indirectamente, incluso a través de funciones). Los elementos de origen y los resultados de las funciones se añaden al contexto para este nodo.
- Por cada nodo de destino nuevo se establece un contexto nuevo, que inicialmente contiene todos los elementos del contexto del nodo primario. Los nodos de destino del mismo nivel, por tanto, son independientes entre sí pero tienen acceso a todos los datos de origen de sus nodos primarios.

Por ejemplo, en el caso de la asignación (**PersonListByBranchOffice.mfd**):

- La conexión que une a **Office** con **PersonList** a través del filtro (Office) define una sola oficina como contexto para todo el documento de destino (porque PersonList es el elemento raíz del componente de destino). El nombre de la oficina viene dado por el componente de entrada, que tiene el valor predeterminado "Nanonull, Inc."
- Todos los datos/todas las conexiones de los **descendientes** del elemento raíz PersonList se ven afectados automáticamente por la condición de filtrado porque la oficina seleccionada está en el contexto.
- La conexión que une a **Contact** con **Person** crea un Person de destino por cada elemento Contact del XML de origen (la regla general se aplica). Por cada Person se añade un Contact concreto al contexto, a partir del cual se crearán los secundarios de Person.
- El conector que une a **first** con **First** selecciona el nombre del Contact actual y lo escribe en el elemento First de destino.

Si se obvia el conector que une **Contact** con **Person**, se crearía solo un Person con varios nodos First, Last y Detail, pero eso no es lo que queremos en este caso. Así que cuando eso ocurre MapForce emite una advertencia y una sugerencia para solucionar el problema: "Puede intentar conectar Contact con Person para resolverlo.":



Secuencias

MapForce muestra la estructura de un archivo de esquema en el componente como una jerarquía de elementos que se pueden asignar.

Dependiendo de cuál sea el **contexto (de destino**), cada elemento asignable de un componente de origen puede representar:

- un **solo nodo de instancia** del archivo de entrada asignado
- una **secuencia de cero a varios nodos de instancia** del archivo de entrada

Si hay una secuencia conectada a un nodo de destino, se crea un bucle para crear tantos nodos de destino como nodos de origen existan.

Si se coloca un filtro entre la secuencia y el nodo de destino, se verifica la condición binaria por cada nodo de entrada (es decir, por cada elemento de la secuencia). Más concretamente se comprueba si en cada secuencia hay como mínimo un booleano que dé true como resultado. La configuración de contexto prioritario puede influir en el orden de evaluación (ver más abajo).

Como se dijo anteriormente, las condiciones de filtrado afectan automáticamente a todos los nodos descendientes.

Nota: si el esquema de origen especifica que determinado nodo aparece una sola vez, MapForce puede eliminar el bucle y tomar solo el primer elemento, que sabe que existe. Esta optimización se puede deshabilitar en el cuadro de diálogo "Configuración del componente" (casilla *Optimización de procesamiento de datos de entrada basada en min/maxOccurs"*).

Las **entradas de las funciones** (normales, no secuenciales) funcionan de forma similar: si una secuencia está conectada a una entrada de función, se crea un bucle alrededor de la llamada a función para que produzca **tantos resultados** como elementos hay en la secuencia.

Si una secuencia está conectada a más de una entrada de función, MapForce crea bucles anidados que procesarán el **producto cartesiano** de todas las entradas. Esto no suele ser lo más indicado, así que no conecte más de una secuencia formada por varios elementos a una función.

Nota: si hay una secuencia vacía conectada a una función (p. ej. `concat`), recibirá como resultado una secuencia vacía, que no producirá ningún nodo de destino. Si no hay resultados en los documentos de salida porque no hay datos de entrada, puede usar la función `substitute-missing` para insertar valores sustitutos.

Las funciones con **entradas secuenciales** son las únicas funciones que pueden producir un resultado si la secuencia de entrada está **vacía**:

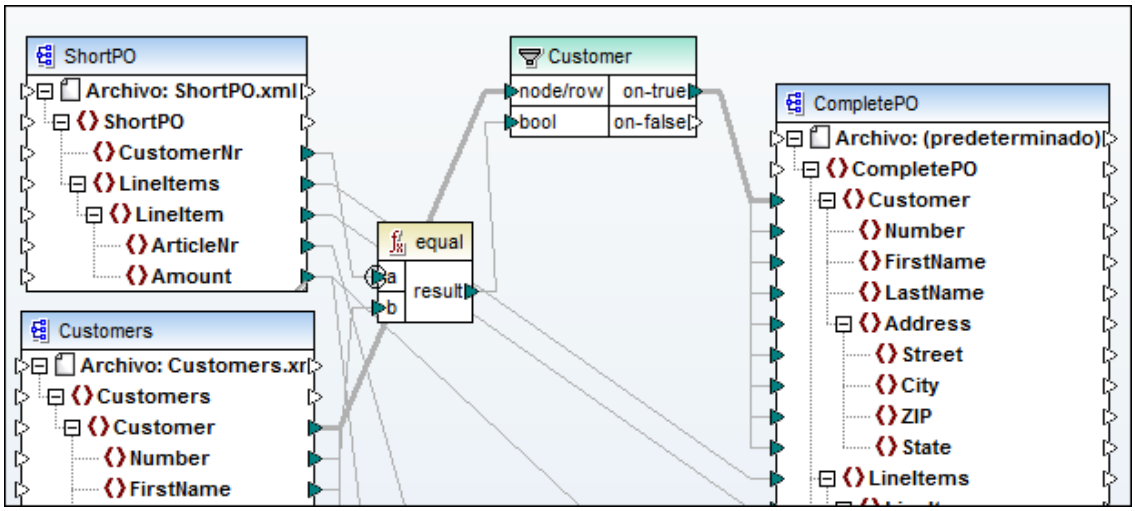
- **`exists`**, **`not-exists`** y **`substitute-missing`** (también, **`is-not-null`**, **`is-null`** y **`substitute-null`**, que son los alias para las tres primeras)
- funciones de agregado (**`sum`**, **`count`**, etc.)
- funciones definidas por el usuario que acepten secuencias (es decir, funciones no insertadas).

La entrada secuencia de dichas funciones siempre se evalúa independientemente del nodo de destino actual en el contexto de sus antecesores. Esto también significa que cualquier componente de filtrado que esté conectado a dichas funciones no tendrán efecto alguno en las demás conexiones.

Contexto prioritario

Por lo general los parámetros de función se evalúan de arriba a abajo, pero también se puede definir que un parámetro se evalúe antes que los demás. Esto se hace estableciendo un **contexto prioritario**.

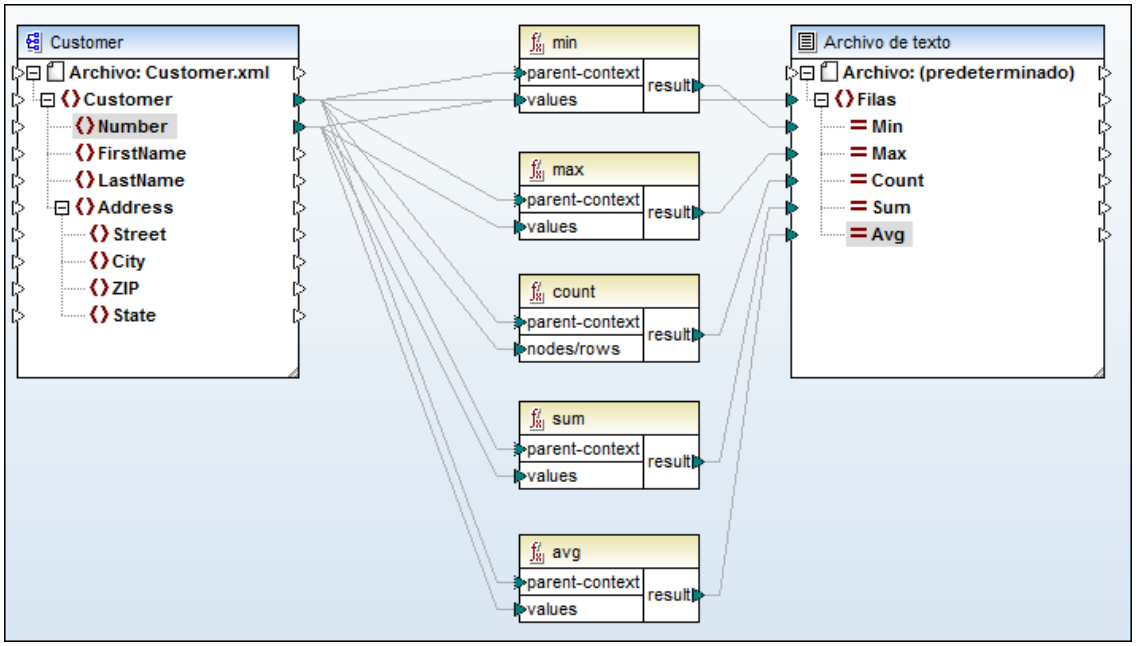
En las funciones que están conectadas a la entrada bool de condiciones de **filtrado**, el contexto prioritario no solo afecta a la función de comparación sino a la evaluación del filtro, así que se pueden unir dos secuencias de origen (*véase CompletePO.mfd: CustomerNo y Number*).



En este ejemplo, el contexto prioritario fuerza a evaluar ShortPO/CustomerNr antes de recorrer y de filtrar los nodos de Customer del componente Customers. Consulte el apartado Nodo de contexto prioritario para obtener más información.

Reemplazar el contexto

Algunas funciones de agregado tienen una entrada opcional llamada “parent-context”. Si esta entrada se deja sin conectar, no tiene efecto alguno y la función se evalúa en el contexto normal de para entradas secuencias (es decir, en el contexto del primario del nodo de destino).



Si por el contrario se conecta la entrada `parent-context` a un nodo de origen, la función se evalúa por cada nodo `parent-context` y producirá un resultado distinto por cada instancia. Consulte el apartado Reemplazar el contexto de la asignación para obtener más información.

Traer varios nodos del mismo componente de origen al contexto

Esto puede ser necesario en algunos casos y se puede hacer con ayuda de variables intermedias.

5.12.1 Cambiar el orden de procesamiento de los componentes

En MapForce pueden diseñarse asignaciones con varios componentes de destino. Cada uno de ellos cuenta con un botón de vista previa que permite consultar el resultado que genera dicho componente.

Si la asignación se ejecuta desde la línea de comandos o desde el código generado, entonces se ejecuta toda la asignación (independientemente de qué botón de vista previa esté activo) y se genera el resultado de todos los componentes de destino.

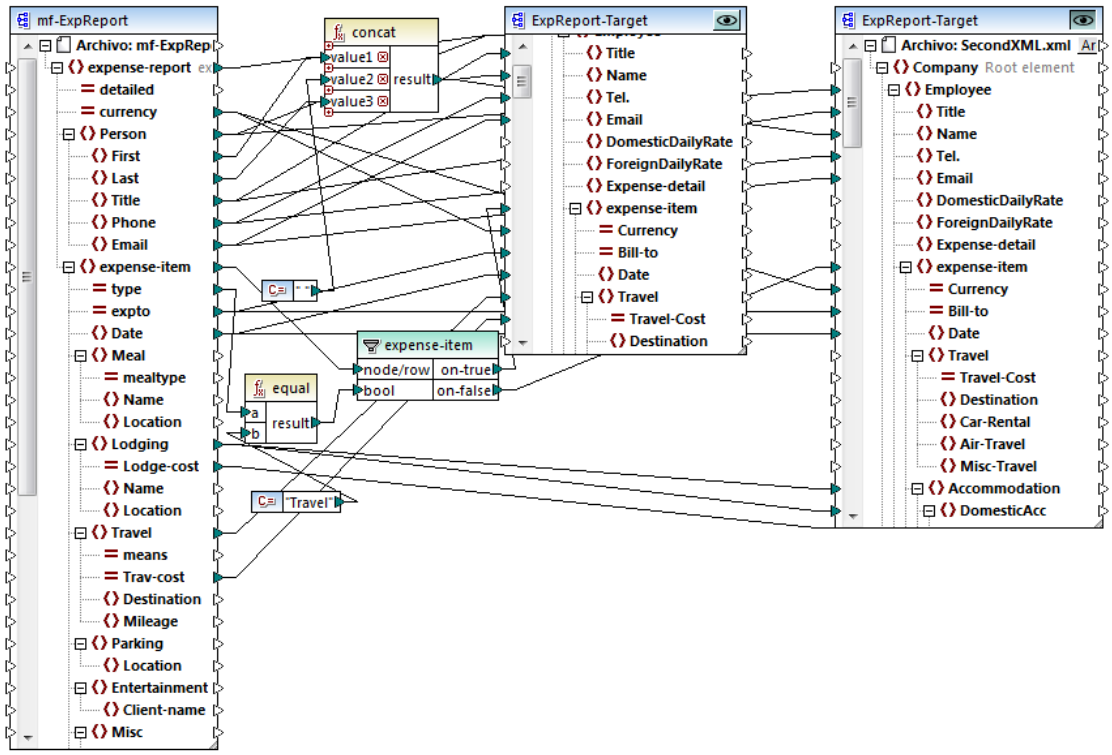
El orden en el que se procesan los componentes de destino se puede modificar con solo cambiar la posición de los componentes de destino. La posición de un componente viene definida por la posición de su **esquina superior izquierda**.

Los componentes de destino se procesan de abajo a arriba y de izquierda a derecha:

- Si hay dos componentes con la misma posición vertical, se procesa primero el situado a la izquierda.
- Si hay dos componentes con la misma posición horizontal, se procesa el situado más arriba.
- En el caso improbable de que dos componentes tengan una posición idéntica, se usa

automáticamente un identificador de componente interno y único que garantiza un orden de procesamiento bien definido, pero inalterable.

En la imagen siguiente aparece el tutorial **Tut-ExpReport-multi.mfd**, situado en la carpeta **<Documentos>\Altova\MapForce2018\MapForceExamples\Tutorial**, que tiene dos componentes de destino. Ambos componentes tienen la misma posición **vertical** y el componente de destino de la derecha tiene activado el botón de vista previa.



Tut-ExpReport-multi.mfd (MapForce Enterprise Edition)

Si seleccionamos XSLT2 y generamos el código:

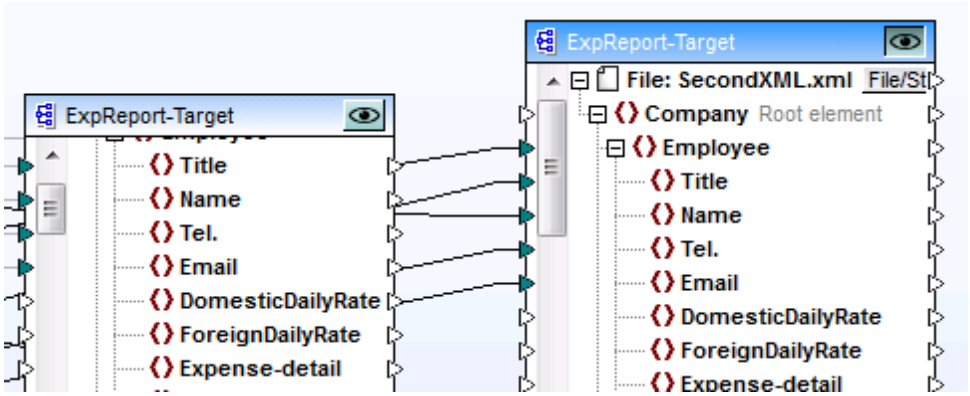
- Primero se procesa el componente de destino de la izquierda y se genera el archivo **ExpReport.xml**.
- Después se procesa el componente de destino de la derecha y se genera el archivo **SecondXML.xml**.

Para comprobar si éste es el orden de procesamiento abra el archivo **DoTransform.bat** (situado en la carpeta de salida que se especificara) y consulte la secuencia de los archivos de salida generados. El primer archivo que genera el archivo de procesamiento por lotes es **ExpReport-Target.xml** y el segundo es **SecondXML.xml**.

```
@echo off

RaptorXML xslt --xslt-version=2 --
input="C:\Users\me\Documents\Altova\MapForce2013\MapForceExamples\Tutorial\m
f-ExpReport.xml" --
output="C:\Users\me\Documents\Altova\MapForce2013\MapForceExamples\Tutorial\
ExpReport-Target.xml" %* "MappingMapToExpReport-Target.xslt"
IF ERRORLEVEL 1 EXIT/B %ERRORLEVEL%
RaptorXML xslt --xslt-version=2 --
input="C:\Users\me\Documents\Altova\MapForce2013\MapForceExamples\Tutorial\m
f-ExpReport.xml" --
output="C:\Users\me\Documents\Altova\MapForce2013\MapForceExamples\Tutorial\
SecondXML.xml" %* "MappingMapToExpReport-Target2.xslt"
IF ERRORLEVEL 1 EXIT/B %ERRORLEVEL%
```

Cambiar la secuencia de procesamiento de la asignación:

1. Haga clic en el componente de destino situado a la izquierda y muévalo debajo del situado a la derecha.



2. Vuelva a generar el código y consulte otra vez el archivo **DoTransform.bat**.

```
@echo off

RaptorXML xslt --xslt-version=2 --
input="C:\Users\alp\Documents\Altova\MapForce2013\MapForceExamples\Tutorial\
mf-ExpReport.xml" --
output="C:\Users\alp\Documents\Altova\MapForce2013\MapForceExamples\Tutorial
\SecondXML.xml" %* "MappingMapToExpReport-Target.xslt"
IF ERRORLEVEL 1 EXIT/B %ERRORLEVEL%
RaptorXML xslt --xslt-version=2 --
input="C:\Users\alp\Documents\Altova\MapForce2013\MapForceExamples\Tutorial\
mf-ExpReport.xml" --
output="C:\Users\alp\Documents\Altova\MapForce2013\MapForceExamples\Tutorial
\ExpReport-Target.xml" %* "MappingMapToExpReport-Target2.xslt"
IF ERRORLEVEL 1 EXIT/B %ERRORLEVEL%
```

Ahora el primer archivo que genera el archivo de procesamiento por lotes es **SecondXML.xml** y el segundo es **ExpReport-Target.xml**.

Asignaciones en cadena

La misma secuencia de procesamiento que acabamos de describir es la que se sigue en las asignaciones en cadena. El grupo de la asignación en cadena se entienda como unidad indisoluble y, por tanto, si cambiamos la posición del componente intermedio o de destino final de una asignación en cadena, la secuencia de procesamiento no cambia.

El único caso en el que la posición de los componentes de destino finales de cada grupo determina qué se procesa primero es cuando una asignación tiene varias cadenas o componentes de destino.

- Si dos componentes de destino finales tienen la misma posición vertical, se procesa primero el situado a la izquierda.
- Si dos componentes de destino finales tienen la misma posición horizontal, se procesa primero el situado más arriba.
- En el caso improbable de que dos componentes tengan una posición idéntica, se usa automáticamente un identificador de componente interno y único que garantiza un orden de procesamiento bien definido, pero inalterable.

5.12.2 Nodo de contexto prioritario

Cuando aplicamos una función a distintos elementos del esquema, MapForce necesita saber qué nodo será el nodo de contexto. Los demás nodos se procesan en relación al nodo de contexto. Esto se consigue designando un elemento (o nodo) como contexto prioritario.

El contexto prioritario se utiliza para clasificar por orden de prioridad la ejecución de las asignaciones de los elementos no relacionados entre sí.

Las asignaciones siempre se ejecutan de arriba a abajo. Si recorre/realiza una búsqueda en dos tablas, entonces se procesa primero un bucle y después otro. Cuando se realizan asignaciones de elementos no relacionados entre sí, sin establecer el contexto prioritario, MapForce no sabe qué bucle debe ejecutarse primero. Cuando desconoce el contexto prioritario, MapForce selecciona automáticamente la primera tabla o nodo de origen.

Solución:
Decida en qué datos de origen debe realizarse la búsqueda primero o qué nodo debe recorrerse primero y establezca el contexto prioritario en su conector.

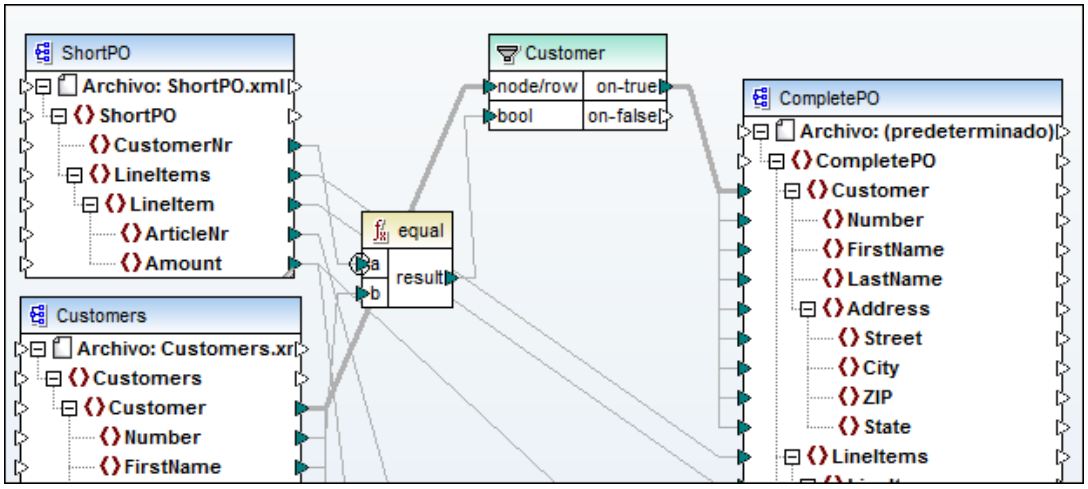
Más abajo puede ver una imagen del archivo **CompletePO.mfd** de la carpeta **...\MapForceExamples**.

Observe que esta asignación tiene varios componentes de origen. **ShortPO, Customers** y **Articles** son esquemas que tienen asociados archivos XML de instancia. Los datos de todos estos componentes se asignan al archivo/esquema XML **CompletePO**. El nodo de contexto prioritario aparece señalado con un círculo.

- El nodo **`CustomerNr`** del esquema de origen **ShortPO** se compara con el elemento **`Number`** del archivo **Customers**.
- **`CustomerNr`** se designa como **contexto prioritario** y se coloca en el parámetro **`a`** de la función **`equal`**.
- En el archivo **Customers** se busca el **mismo** número **una vez**. El parámetro **`b`** contiene el elemento **`Number`** del archivo **Customers**.
- Si se encuentra el número, el resultado se pasa al parámetro **`bool`** de la función **`filter`**.
- El parámetro **node/row** pasa los datos de **Customers** a **`on-true`** cuando el parámetro **`bool`** es **`true`**, es decir, cuando se encuentra el mismo número.
- Se pasan los demás datos de **Customers** (elementos **`Number`**, **`FirstName`**, **`LastName`**) y todos se conectan a los elementos correspondientes del esquema de destino.

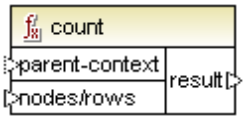
Al designar como **contexto prioritario** el parámetro **b** de la función **equal** (es decir, el elemento **Number**) conseguimos esto:

- MapForce carga el primer **Number** en el parámetro **b**
- Lo compara con el elemento **CustomerNr** del parámetro **a**
- Si no son iguales, MapForce carga el siguiente **Number** en el parámetro **b** y lo comprueba con **a**
- Se itera en cada elemento **Number** del archivo y se intenta buscar ese número en **ShortPO**.



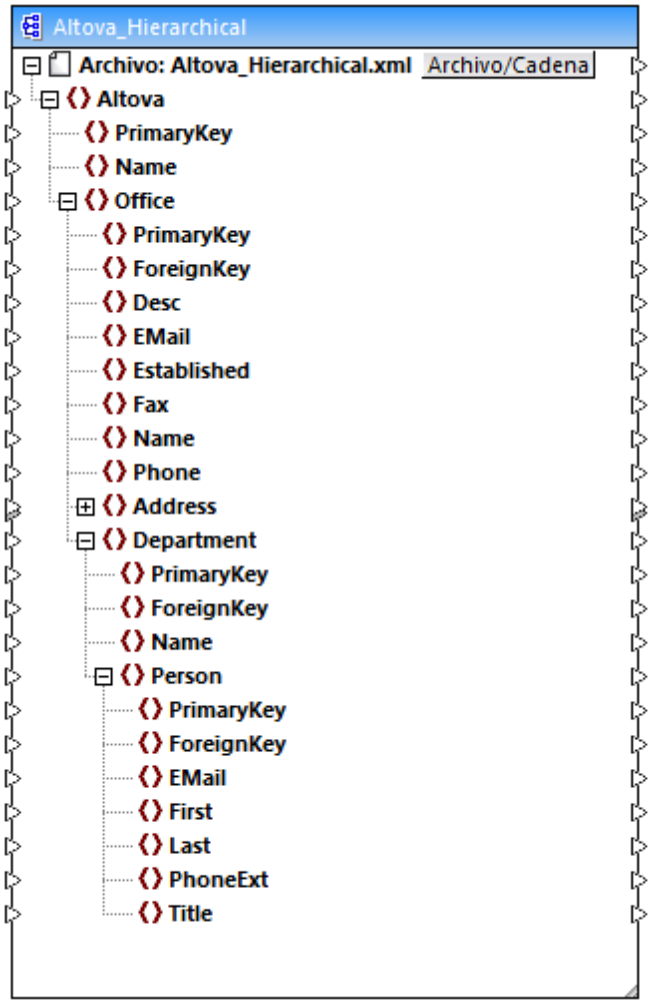
5.12.3 Reemplazar el contexto de la asignación

En algunas asignaciones, para poder conseguir el resultado deseado, puede ser necesario reemplazar el contexto de la asignación. Por este motivo la estructura de algunos componentes ofrece un elemento opcional llamado `parent-context` que permite determinar el contexto de la asignación. Por ejemplo, las funciones de agregado y las variables cuentan con el elemento opcional `parent-context`.



Función de agregado con el elemento opcional parent-context

Para entender lo importante que es el contexto de la asignación añadiremos a la asignación un archivo XML que contiene nodos anidados con varios niveles. En el menú **Insertar** haga clic en el comando **Archivo o esquema XML** y navegue hasta el archivo **<Documentos>\Altova \MapForce2018\MapForceExamples\Altova_Hierarchical.xml**.



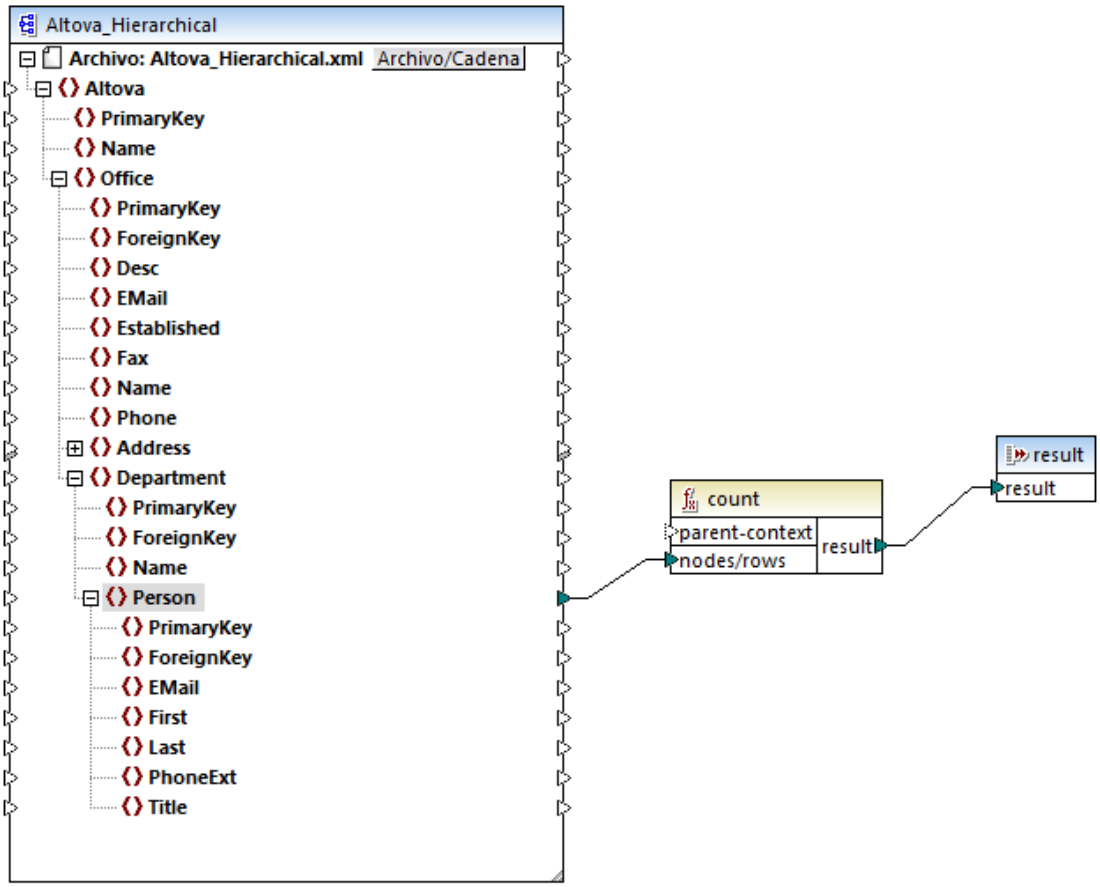
Altova_Hierarchical.xml

Lo más importante es observar que, en el archivo XML, el nodo primario `Office` contiene varios nodos `Department` que a su vez contienen varios nodos `Person`. Si abrimos el archivo XML en un editor XML podremos ver la distribución de empleados por oficina y por departamento:

Oficina	Departamento	Nº de empleados
Nanonull, Inc.	Administration	3
	Marketing	2
	Engineering	6
	IT & Technical Support	4
Nanonull Partners, Inc.	Administration	2

Oficina	Departamento	Nº de empleados
	Marketing	1
	IT & Technical Support	3

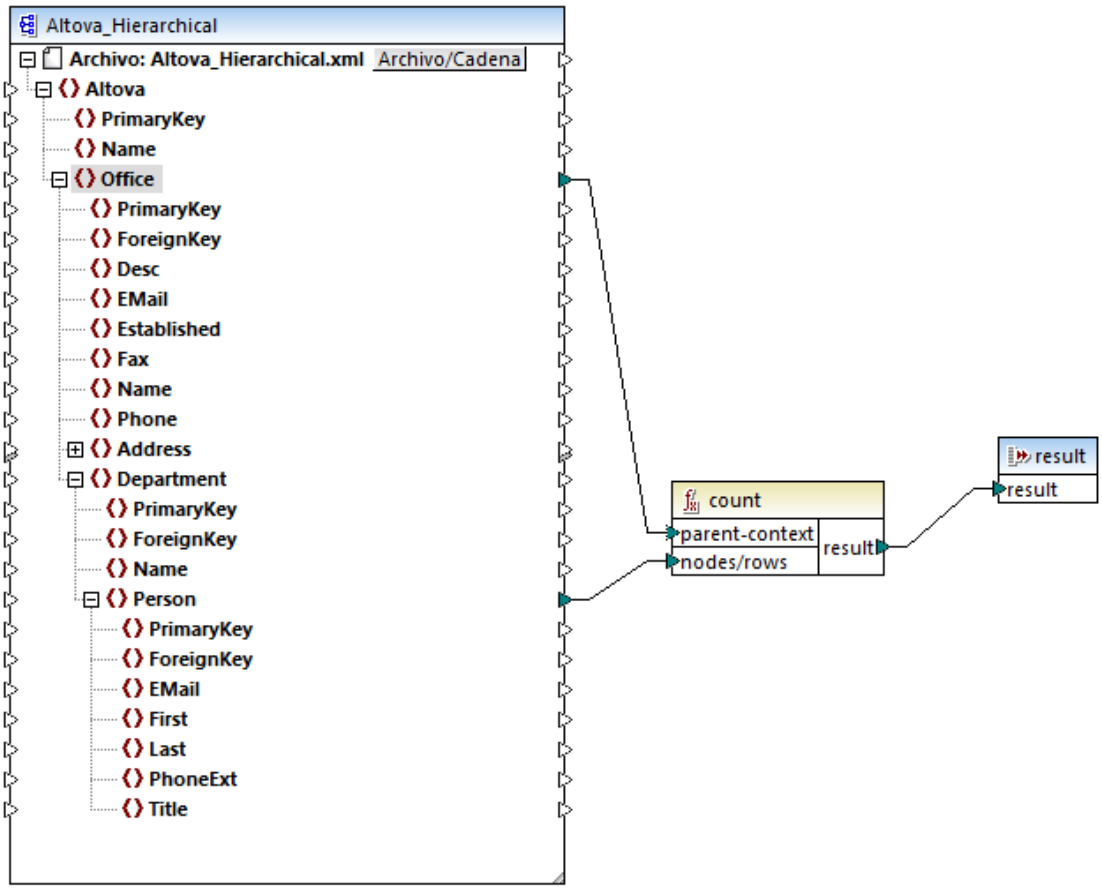
Ahora imaginemos que la asignación debe contar todos los empleados de todos lo departamentos. Para conseguir este objetivo podemos añadir la función **count** de la biblioteca de funciones core | aggregate functions y crear estas asignaciones de datos:



Si consultamos la vista previa del resultado en este momento, veremos que el resultado es `21`, es decir, el número total de empleados de todos los departamentos juntos. Observe que la función **count** tiene un elemento opcional llamado `parent-context` que no hemos utilizado por ahora. Como resultado, el contexto primario de la función **count** es el nodo raíz predeterminado del componente de origen (que, en este caso, es el elemento `Altova`). Esto significa que se tienen en cuenta todos los empleados de todos los departamentos para el ámbito de la función **count**. Este es el funcionamiento predeterminado del contexto de la asignación (*véase Reglas y estrategias de asignación de datos*) y suele ser el funcionamiento adecuado para la mayoría de asignaciones.

Sin embargo, en MapForce se puede reemplazar el contexto predeterminado de la asignación.

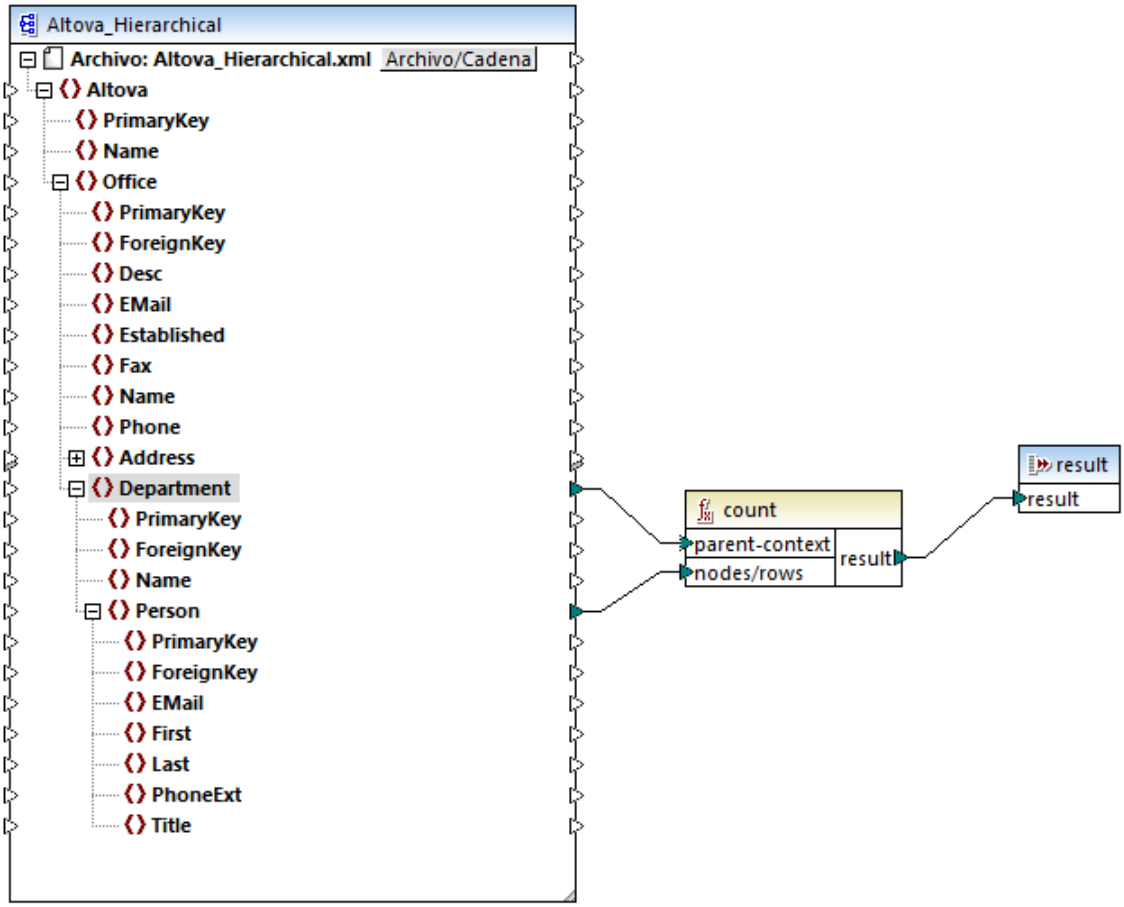
Esto se hace creando una conexión entre el nodo `Office` y el elemento opcional `parent-context` de la función **`count`**.



Al crear esta conexión estamos indicando que la asignación debe recorrer los registros de empleados *en el contexto de cada oficina*. Es decir, si ahora consultamos la vista previa del resultado de la asignación, el resultado será `15*`. Se trata del número de empleados de la primera oficina ("Nanonull, Inc."). El motivo es que esta vez los nodos `Person` se contaron dos veces (una vez por cada oficina). El número de empleados de las oficinas es 15 y 6 respectivamente, pero solamente se devuelve el primer resultado (porque la función no puede devolver una secuencia de valores sino un solo valor).

** Siempre y cuando el lenguaje de destino elegido para la asignación no sea XSLT 1.0.*

Puede seguir modificando la asignación y su contexto. Por ejemplo, ahora cambiamos el contexto por `Department` (*imagen siguiente*). Esta vez los registros de empleados se contarán en el contexto de cada departamento (es decir, 7 veces, tantas como departamentos). Una vez más, solamente se devuelve el primero de los resultados. Así que el resultado de la asignación es `3`, que es el número de empleados del primer departamento de la primera oficina.



Aunque esta asignación no es de gran utilidad, nos sirve para explicar cómo influye el elemento opcional `parent-context` en el resultado de la asignación. Ahora puede tener esto en cuenta y reemplazar el contexto primario `parent-context` en sus asignaciones (p. ej. en las que contienen variables.

Temas relacionados: *Ejemplo: crear grupos y subgrupos de registros*

Altova MapForce 2018 Basic Edition

Orígenes y destinos de datos

6 Orígenes y destinos de datos

En esta sección describimos todos los tipos de componente de origen y destino compatibles con MapForce.

6.1 XML y esquemas XML

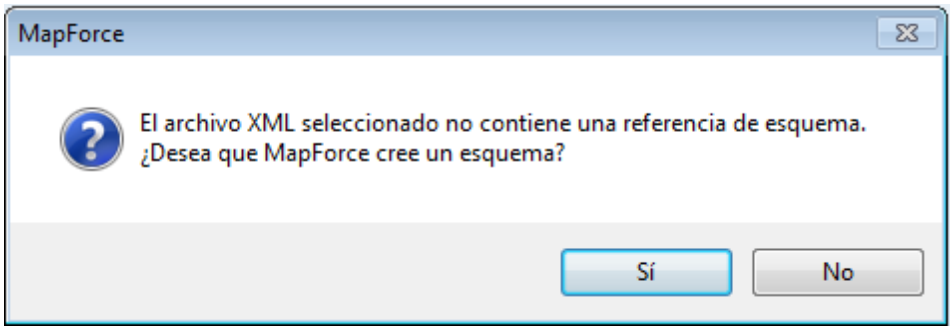
Sitio web de Altova: Asignaciones XML

En los temas de introducción de esta documentación se presentaron ejemplos de asignaciones de datos simples que utilizaban archivos XML y de esquema XML como componentes de origen y destino. En esta sección encontrará más información sobre este tipo de componentes, organizada por temas:

- Configuración de componentes XML
- Usar documentos DTD como componentes de esquema
- Asignación de datos de tipos XML Schema derivados
- Compatibilidad con nombres QName
- Valores Nil / Nillable
- Comentarios e instrucciones de procesamiento
- Secciones CDATA
- Comodines xs:any

6.1.1 Generar un esquema XML

MapForce puede generar automáticamente un esquema XML a partir de un archivo XML. Cada vez que se añade un archivo XML que carece de esquema al área de asignación (con el comando **Insertar | Archivo o esquema XML**), MapForce muestra este mensaje:



Haga clic en **Sí** para generar el esquema. Después deberá seleccionar el directorio donde desea guardar el esquema resultante.

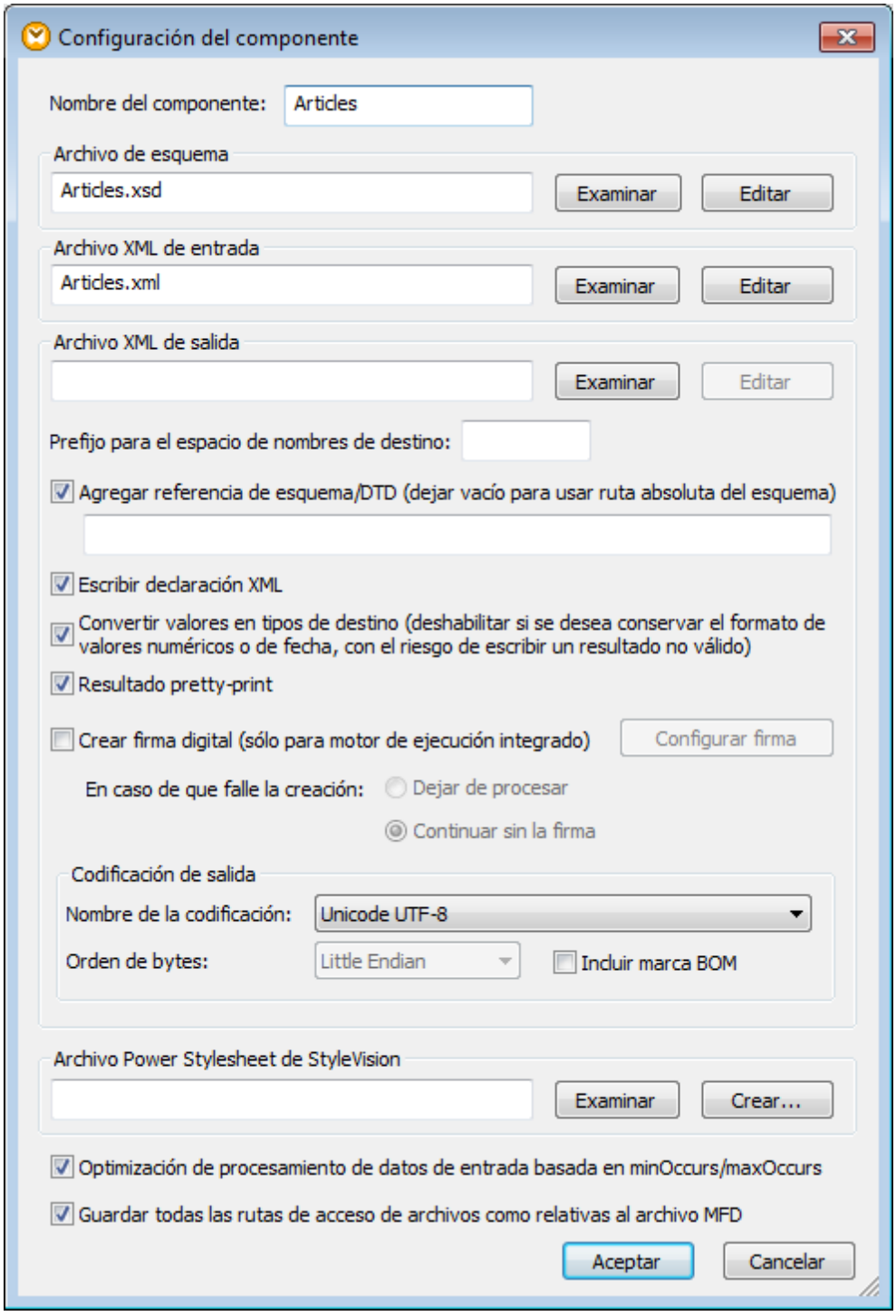
Cuando MapForce genera un esquema a partir de un archivo XML, los tipos de datos para elementos y atributos deben inferirse del documento de instancia XML y puede que el resultado no sea exactamente lo que se esperaba. Por eso, recomendamos que terminada la operación compruebe si el esquema generado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

6.1.2 Configuración de componentes XML

Tras añadir un componente XML al área de asignación, podrá configurarlo desde el cuadro de diálogo "Configuración del componente". Este cuadro de diálogo se puede abrir de tres maneras diferentes:

- Seleccionando el componente en el área de asignación y haciendo clic en el comando de menú **Componente | Propiedades**.
- Haciendo doble clic en el título del componente.
- Haciendo clic con el botón derecho en el título del componente y eligiendo **Propiedades** en el menú contextual.



Cuadro de diálogo "Configuración del componente" para un componente XML en MapForce Enterprise y Professional Edition

Esta son las opciones que ofrece el cuadro de diálogo:

Nombre del componente	El nombre del componente se genera automáticamente cuando se crea el componente. Sin embargo, puede cambiar el nombre si lo desea. Si el nombre del componente se generó automáticamente y después selecciona un archivo de instancia, MapForce le preguntará si desea actualizar también el nombre del componente. El nombre del componente puede contener espacios (p. ej. Archivo XML de origen) y puntos (p. ej. Pedidos.EDI). No puede contener barras diagonales, dos puntos, comillas dobles ni espacios iniciales ni finales. Además, cuando cambie el nombre del componente tenga en cuenta que: • Si tiene pensado implementar la asignación en FlowForce Server, el nombre del componente debe ser único. • Se recomienda usar caracteres que se puedan introducir en la línea de comandos. Los caracteres regionales pueden tener distinta codificación en Windows y en la línea de comandos.
Archivo de esquema	Especifica el nombre o la ruta de acceso del archivo de esquema XML que MapForce usará para validar y asignar datos. Para cambiar de archivo de esquema haga clic en **Examinar** y seleccione el nuevo archivo. Para editarlo en XMLSpy haga clic en **Editar**.
Archivo XML de entrada	Especifica el archivo XML de instancia donde MapForce leerá datos. Este campo es relevante para los componentes de origen y se rellena automáticamente cuando se crea el componente por primera vez y se le asigna un archivo XML de instancia. En un componente de origen el nombre del archivo de instancia también se usa para detectar el elemento XML raíz y el esquema al que hace referencia, además de para validar los datos con el esquema seleccionado. Para cambiar la ubicación del archivo haga clic en **Examinar** y seleccione el nuevo archivo. Para editarlo en XMLSpy haga clic en **Editar**.
Archivo XML de salida	Especifica el archivo XML de instancia donde MapForce escribirá los datos. Este campo es relevante para los componentes de destino. Para cambiar la ubicación del archivo haga clic en **Examinar**

<table>
<tr><td></td><td>y seleccione el nuevo archivo. Para editarlo en XMLSpy haga clic en Editar.</td></tr>
<tr><td>Prefijo para el espacio de nombres de destino</td><td>Permite introducir un prefijo para el espacio de nombres de destino. Antes de asignar el prefijo, asegúrese de que en el esquema de destino está definido el espacio de nombres de destino.</td></tr>
<tr><td>Agregar referencia de esquema/DTD</td><td>Agrega la ruta de acceso del esquema XML referenciado al elemento raíz del archivo XML de salida. La ruta de acceso del esquema introducido en este campo se escribe en los archivos de instancia de destino que se generan (en el atributo <code>xsi:schemaLocation</code> o en la declaración <code>DOCTYPE</code> si se usa una DTD.

Al introducir una ruta de acceso en este campo podrá definir dónde está ubicado el archivo de esquema al que hace referencia el archivo XML de instancia. Esto garantiza que la instancia de salida se pueda validar en el destino de la asignación cuando se ejecute la asignación. En este campo puede introducir tanto una dirección http:// como una ruta de acceso absoluta o relativa.

Si desactiva esta opción, podrá desvincular la instancia XML del esquema XML o DTD de referencia (por ejemplo, si desea enviar el documento XML de salida a alguien que no tiene acceso al esquema XML subyacente).</td></tr>
<tr><td>Escribir declaración XML</td><td>Esta opción permite suprimir la declaración XML del documento de salida que se genera. Esta opción está habilitada por defecto, es decir, la declaración XML se describe por defecto en los resultados.

Esta característica no es compatible con todos los lenguajes de destino y motores de ejecución de MapForce, como puede verse en la tabla siguiente.

<table>
<tr><th>Lenguaje de destino / motor de ejecución</th><th>Si el resultado es un archivo</th><th>Si el resultado es una cadena de texto</th></tr>
<tr><td>XSLT, XQuery</td><td>Sí</td><td>No</td></tr>
</table>
</td></tr>
<tr><td>Convertir valores en tipos de destino</td><td>Permite definir si los tipos del esquema XML de destino deben utilizarse durante la asignación o si todos los datos asignados al componente de destino deben considerarse valores de cadena. Esta opción está habilitada por defecto.

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.</td></tr>
</table>

	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 desplaza una tabulación con respecto a su primario.
Codificación de salida	Permite especificar estas opciones de configuración para el archivo de instancia de salida: • Nombre de la codificación • Orden de bytes • Si se incluye o no la marca BOM Por defecto, todos los componentes nuevos tienen la codificación definida en la opción *Codificación predeterminada para componentes nuevos*. Esta opción se puede modificar en la pestaña *Generales* de **Herramientas | Opciones**. Si la asignación de datos genera código XSLT 1.0/2.0, el estado de la casilla *Incluir marca BOM* no tiene ningún efecto porque estos lenguajes no son compatibles con marcas BOM.
Archivo Power Stylesheet de StyleVision	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.
Optimización de procesamiento de datos de entrada basada en minOccurs/maxOccurs	Esta opción permite tratar de forma especial las secuencias que contienen un elemento exactamente, como por ejemplo los atributos obligatorios o los elementos secundarios con atributos `minOccurs` y `maxOccurs="1"`. En este caso, MapForce extrae el primer elemento de la secuencia y después procesa el elemento directamente como valor atómico (y no como secuencia). Si los datos de entrada **no son válidos** según el esquema, puede que la asignación incluya una secuencia vacía que detenga la ejecución con un mensaje de error. Para permitir el procesamiento de dicha **entrada no válida** basta con desactivar esta casilla.

Guardar todas las rutas de acceso de archivos como relativas al archivo MFD	Cuando se habilita esta opción, MapForce guarda las rutas de acceso de archivos que aparecen en el cuadro de diálogo "Configuración del componente" como relativas a la ubicación del archivo de diseño de MapForce (.mfd). Consulte también el apartado Usar rutas de acceso relativas en un componente.

6.1.3 Usar documentos DTD como componentes de esquema

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.

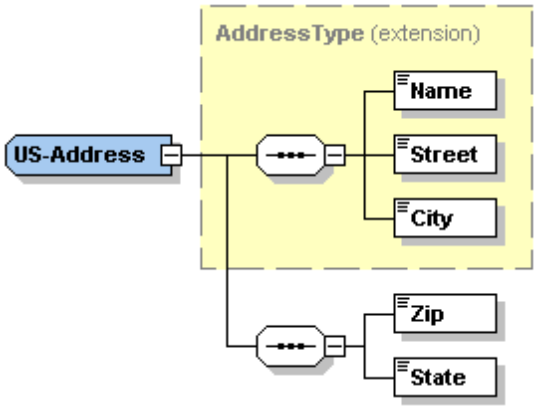
Sin embargo, algunas DTD contienen declaraciones de atributo "xmlns" sin URI de espacio de nombres (p. ej. las DTD que utiliza StyleVision). Estas DTD se deben ampliar para poder utilizarse en MapForce. Concretamente los atributos "xmlns" deben definirse con el URI de espacio de nombres tal y como aparece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

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

6.1.4 Tipos XML Schema 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.

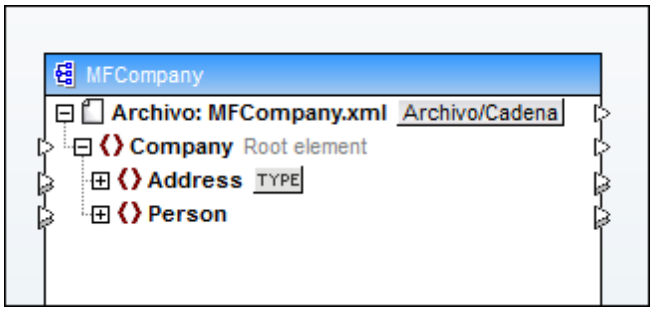
En la imagen siguiente puede ver la definición de un tipo derivado llamado `US-Address` (en XMLSpy). El tipo base (o tipo complejo originario) es `AddressType`. Se añadieron también dos elementos más para crear los tipos derivados `US-Address`: `Zip` y `State`.



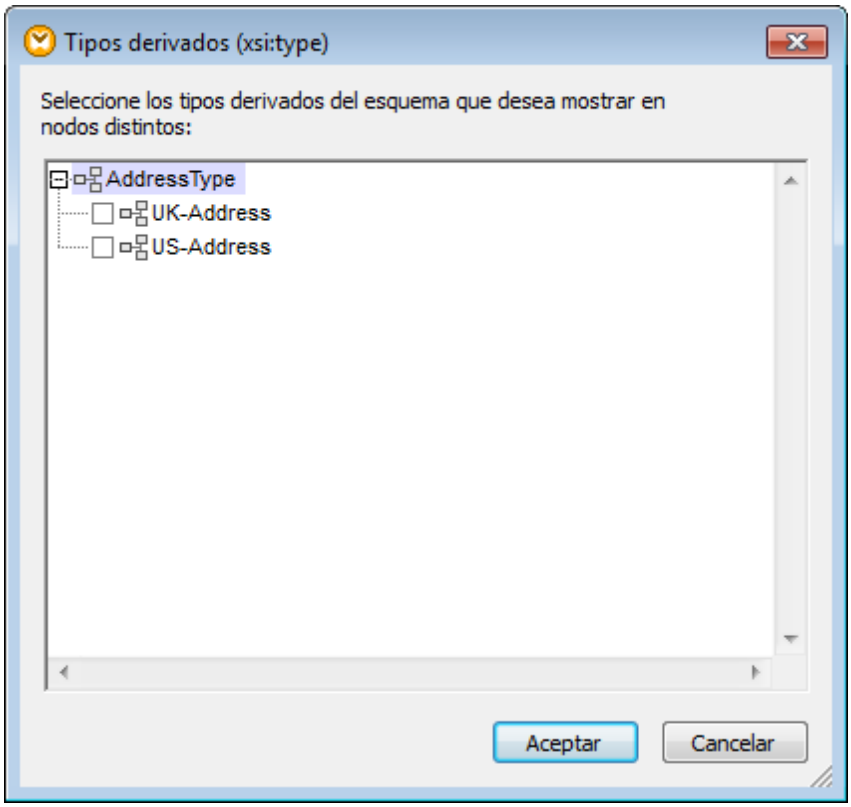
Tipo derivado de muestra (en la vista Esquema de XMLSpy)

A continuación explicamos cómo realizar asignaciones de datos entre tipos derivados de esquema XML:

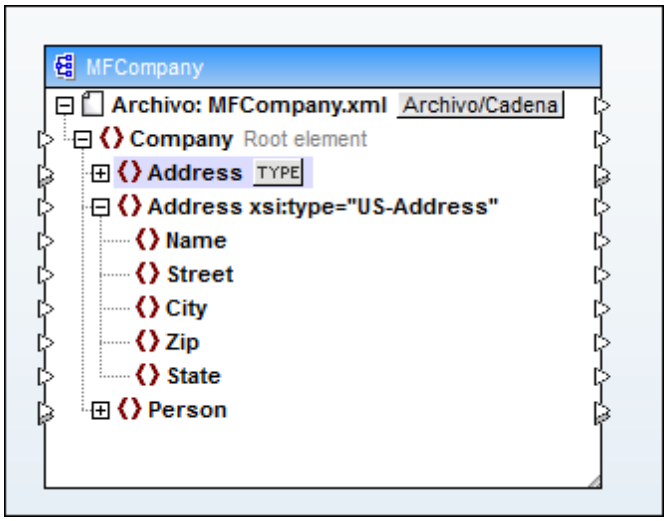
1. En el menú **Insertar** haga clic en el comando **Archivo o esquema XML** y navegue hasta este esquema XML: **<Documentos>\Altova\MapForce2018\MapForceExamples\Tutorial\MFCompany.xsd**.
2. Cuando MapForce solicite un archivo de instancia haga clic en **Omitir** y seleccione el elemento raíz `Company`.



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. Marque la casilla del tipo derivado que desea utilizar (en este caso `US-Address`) y haga clic en **Aceptar** para confirmar. Al componente se añade un elemento nuevo llamado `Address xsi:type="US-Address"`.



Ahora puede crear asignaciones de datos entre el tipo derivado `US-Address` y otros elementos.

Recuerde que en el cuadro de diálogo "Tipos derivados" puede seleccionar varios tipos derivados para incluirlos en el componente. En este caso, cada uno de los tipos derivados tendrían su propio elemento `xsi:type` en el componente.

6.1.5 Nombres QName

MapForce resuelve los prefijos de nombres QName (nombre completo) (http://www.w3.org/TR/xml-names/#ns-qualnames) cuando lee datos de archivos XML en tiempo de ejecución de la asignación.

Los nombres QName se usan para hacer referencia a identificadores URI de espacio de nombres de documentos de instancia XML y para abreviarlos. Hay dos tipos de QName: con prefijo y sin prefijo:

NombreConPrefijo	Prefijo ':'	ParteLocal
NombreSinPrefijo		ParteLocal

siendo ParteLocal un nombre de elemento o atributo.

Por ejemplo, en el fragmento de código que aparece a continuación `<x:p/>` es un QName:

- el prefijo "x" es una abreviatura del espacio de nombres "**http://miCompañía.com**".
- `p` es la parte local.

```
<?xml version='1.0'?>
<doc xmlns:x="http://miCompañía.com">
   <x:p/>
</doc>
```

6.1.6 Valores Nil y Nillable

La especificación XML Schema permite que un elemento sea válido sin tener contenido si el atributo `nillable="true"` se definió para dicho elemento en el esquema. En el documento de instancia XML se puede indicar después que el valor de un elemento es nulo añadiéndole el atributo `xsi:nil="true"`. En este apartado describimos cómo se ocupa MapForce de estos elementos en los componentes de origen y destino.

Diferencias entre 'xsi:nil' y 'nillable'

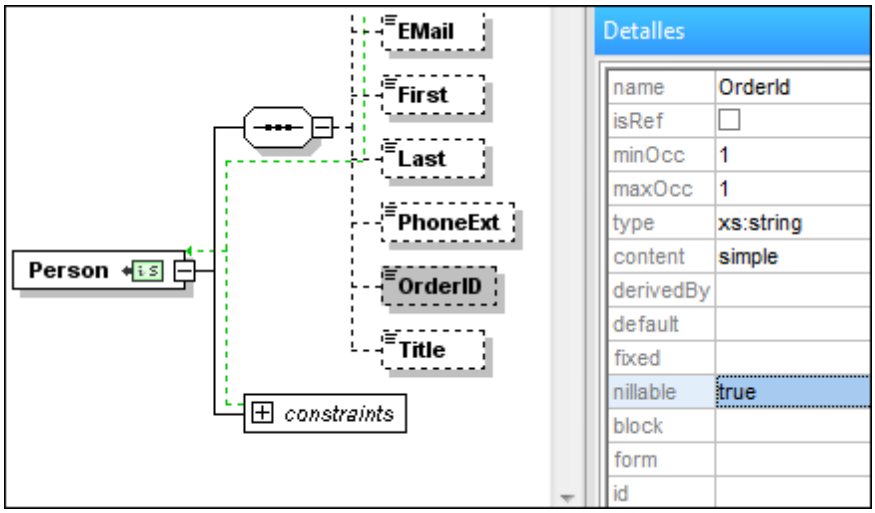
El atributo `xsi:nil="true"` se define en el documento XML de **instancia**.

```
14  <Person>
15    <PrimaryKey>2</PrimaryKey>
16    <ForeignKey>1</ForeignKey>
17    <EMail>biff@amail.com</EMail>
18    <First>biff</First>
19    <Last>bander</Last>
20    <PhoneExt>22</PhoneExt>
21    <OrderID xsi:nil="true"/>
22    <Title>IT services</Title>
23  </Person>
```

El atributo `xsi:nil="true"` indica que, aunque exista, el elemento no tiene contenido. Recuerde que el atributo `xsi:nil="true"` afecta al valor de los elementos y no de los atributos. Un elemento con `xsi:nil="true"` puede tener otros atributos aunque no tenga contenido.

El atributo `xsi:nil` no se representa de forma explícita en la asignación gráfica de MapForce porque se procesa automáticamente en la mayoría de los casos. En concreto, existe un nodo con valor nulo (es decir, un nodo que tiene el atributo `xsi:nil="true"`) pero su contenido no existe.

El atributo `nillable="true"` se define en el **esquema** XML. En MapForce puede estar presente tanto en el componente de origen como en el de destino.



Elementos que pueden ser nulos como origen de la asignación

MapForce revisa automáticamente el atributo `xsi:nil` si la asignación lee datos de elementos XML con valor nulo. Si el valor de `xsi:nil` es `true`, el contenido se tratará como inexistente.

Cuando se crea una asignación **basada en el destino** entre un elemento de origen que puede ser nulo y un elemento de destino que puede ser nulo con **contenido simple** (un solo valor con atributos opcionales, pero sin elementos secundarios), donde se definió `xsi:nil` en un elemento de origen, MapForce añade el atributo `xsi:nil` al elemento de destino (p. ej. `<OrderID xsi:nil="true"/>`).

Cuando se crea una asignación de **copia total** entre un elemento de origen que puede ser nulo y un elemento de destino que puede ser nulo, donde se definió `xsi:nil` en un elemento de origen, MapForce añade el atributo `xsi:nil` al elemento de destino (por ejemplo, `<OrderID xsi:nil="true"/>`).

Para comprobar de forma explícita si un elemento de origen tiene un atributo `xsi:nil` con valor `true`, use la función **`is-xsi-nil`**. La función devuelve TRUE para los elementos con valor nulo y FALSE para los demás nodos.

Para sustituir el valor de un elemento de origen nulo (inexistente) con un valor concreto use la función **`substitute-missing`**.

Notas:

- Si se conecta a un elemento de origen con valor nulo, la función **exists** devuelve TRUE porque el nodo de elemento existe aunque no tenga contenido.
- Si se usan en elementos donde se definió `xsi:nil`, las funciones que esperan valores simples (como **multiply** y **concat**) no producen resultados porque no hay ningún contenido de elemento y no se puede extraer ningún valor. Estas funciones se comportan como si el nodo de origen no existiera.

Elementos que pueden ser nulos como destino de la asignación

Cuando se crea una asignación basada en el destino entre un elemento de origen que puede ser nulo y un elemento de destino que puede ser nulo con contenido simple (un solo valor con atributos opcionales pero sin elementos secundarios), donde se definió `xsi:nil` en un elemento de origen, MapForce inserta el atributo `xsi:nil` en el elemento de destino (por ejemplo: `<OrderID xsi:nil="true"/>`). Si el atributo `xsi:nil="true"` no se definió en el elemento XML de origen, entonces el contenido del elemento se asigna al elemento de destino.

Cuando se crea una asignación con un elemento de destino que puede ser nulo con **tipo complejo** (con elementos secundarios), el atributo `xsi:nil` **no** se escribirá automáticamente porque, a la hora de escribir los atributos del elemento, MapForce no puede saber si después vendrán elementos secundarios. En estos casos se recomienda definir una conexión de **copia total** para copiar el atributo `xsi:nil` del elemento de origen.

Cuando se crea una asignación entre una **secuencia vacía** y un elemento de destino, el elemento no se creará, independientemente de si puede ser nulo o no.

Para forzar la creación de un elemento de destino vacío con el atributo `xsi:nil="true"` conecte la función **set-xsi-nil** con el elemento de destino directamente. Esto sirve para elementos de destino con tipo complejo y con tipo simple.

Si el nodo tiene tipo simple, use la función **substitute-missing-with-xsi-nil** para insertar `xsi:nil` en el destino si no hay ningún valor disponible para la asignación. Esto puede ocurrir si el nodo de origen no existe en absoluto o si en un cálculo (en una multiplicación, por ejemplo) interviene un nodo de origen nulo y, por tanto, no se obtuvo ningún resultado.

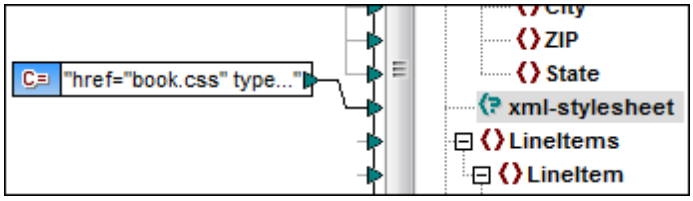
Nota: las funciones que generan `xsi:nil` no pueden pasar por funciones ni componentes que solamente operen en valores (como la función **if-else**, por ejemplo).

6.1.7 Comentarios e instrucciones de procesamiento

En los componentes XML de destino se pueden insertar comentarios e instrucciones de procesamiento. 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.

Para insertar una instrucción de procesamiento:

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**.
2. Introduzca el nombre (de destino) de la instrucción de procesamiento (p. ej. `xml-stylesheet`) y haga clic en **Aceptar** para confirmar.
   Esto añade un nodo de este nombre a la estructura del componente de destino.

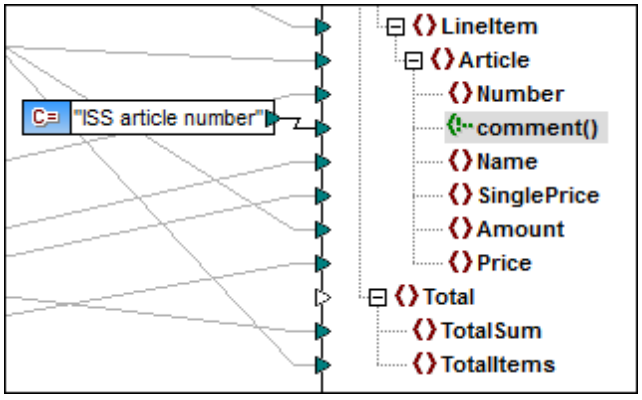


3. Ahora puede usar un componente de constante para aportar el valor del atributo de la instrucción de procesamiento (p. ej. `href="book.css" type="text/css"`).

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

Para insertar un comentario:

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 un comentario** o bien **Agregar detrás un comentario**.



   Esto añade el nodo de comentario (`<!--comment()`) a la estructura del componente de destino.
2. Ahora puede usar un componente de constante para aportar el texto del comentario o conectar un nodo de origen al nodo de comentario.

Nota: solamente se puede añadir un comentario antes o detrás de cada nodo de destino. Para crear varios comentarios deberá crear duplicados de entrada.

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**.

6.1.8 Secciones CDATA

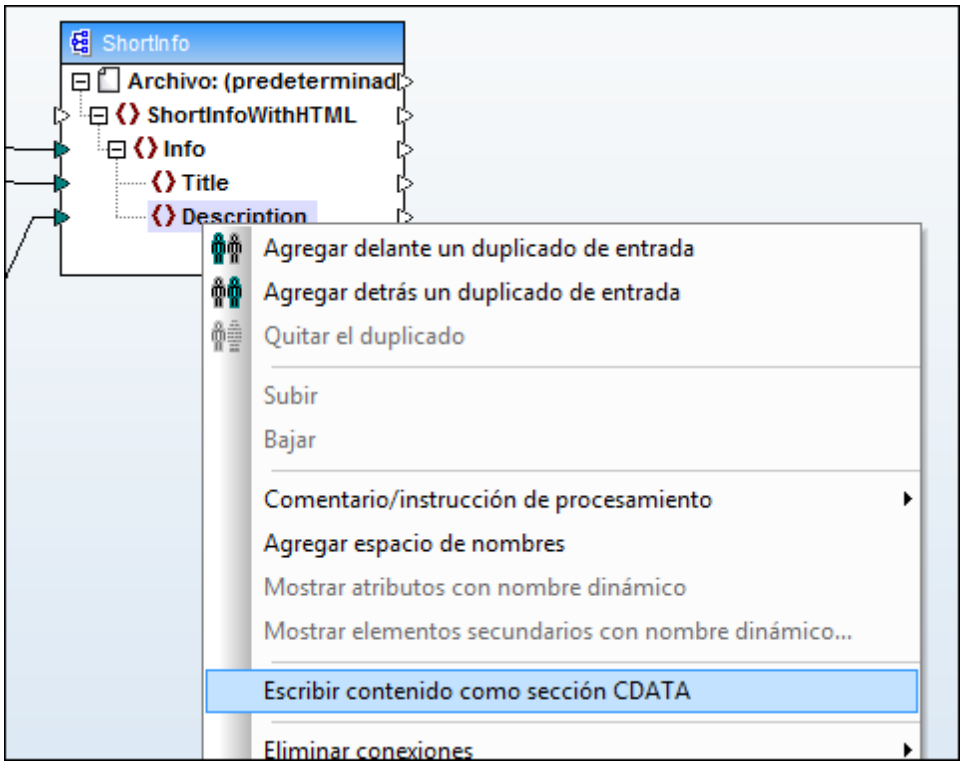
Las secciones CDATA sirven para añadir caracteres de escape a bloques de texto que contienen caracteres que se podrían interpretar como marcado. Las secciones CDATA empiezan con "<![CDATA[" y terminan con "]]>".

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
- elementos secundarios XML de dimensiones con tipo de un componente XBRL de destino

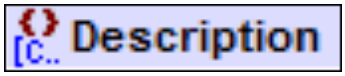
Para crear una sección CDATA:

1. 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.



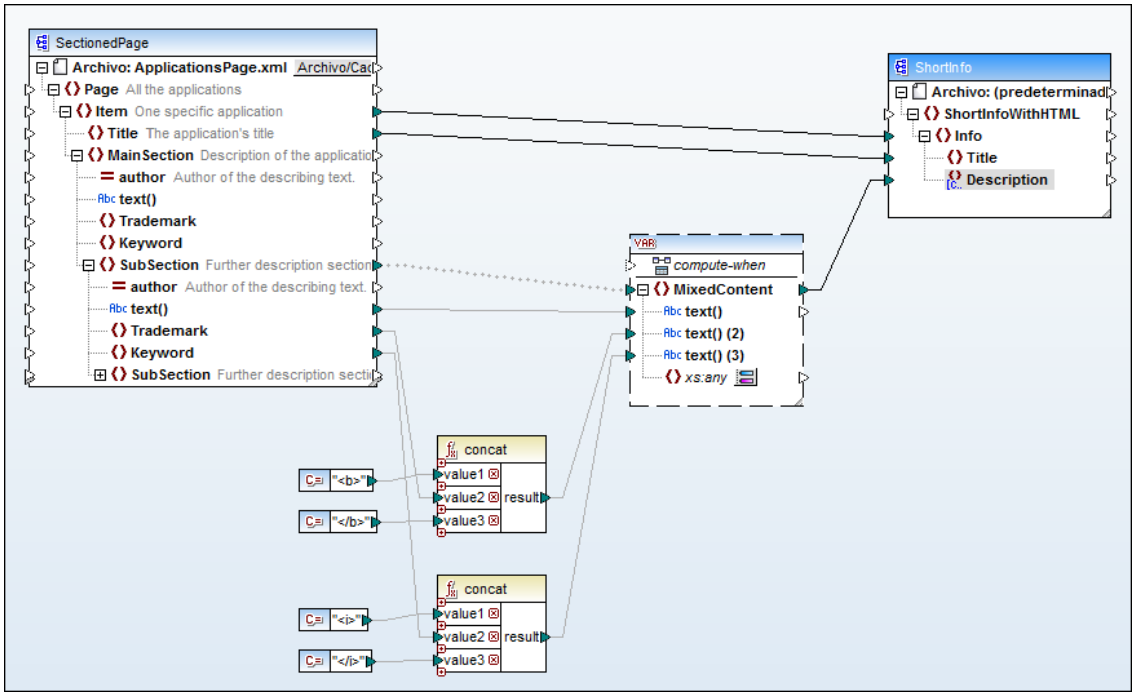
Nota: también se pueden definir secciones CDATA en nodos duplicados y en nodos xsi:type.

Asignación de ejemplo

El diseño de asignación **HTMLinCDATA.mfd** de la carpeta `...\MapForceExamples` muestra lo útil que pueden ser las secciones CDATA.

En este ejemplo:

- Al contenido del elemento de origen **`Trademark`** se añaden las etiquetas de apertura y cierre y .
- Al contenido del elemento de origen **`Keyword`** se añaden las etiquetas de apertura y cierre y
- 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.

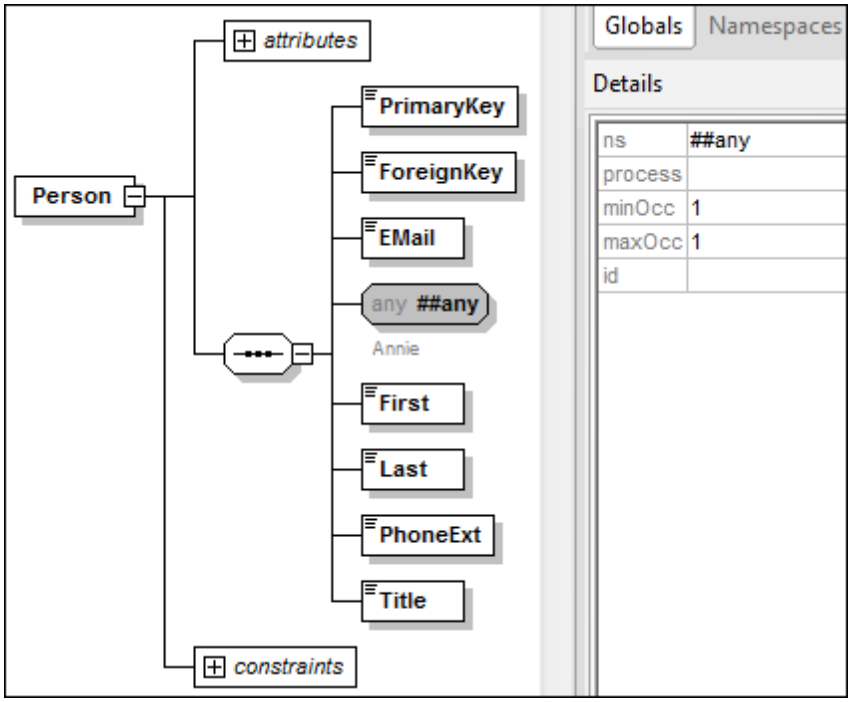


Si abrimos el panel *Resultados* podremos ver la sección CDATA que contiene el texto con marcado.

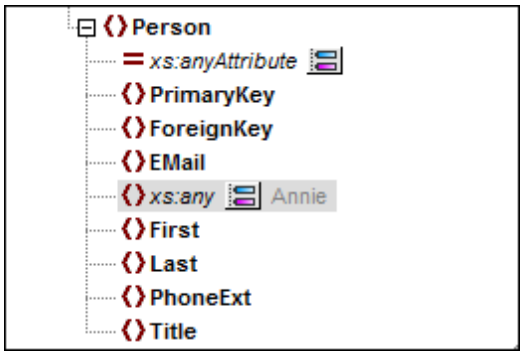
```
7   <Info>
8     <Title>MapForce</Title>
9     <Description><![CDATA[Altova <b>MapForce</b> 2014 Enterprise Edition is the premier <i>XML</i>
/ <i>database</i> / <i>flat file</i> / <i>EDI</i> data mapping tool that auto-generates mapping code in
<i>XSLT</i> 1.0/2.0, <i>XQuery</i>, <i>Java</i>, <i>C++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

6.1.9 Comodines: xs:any / xs:anyAttribute

Los comodines `xs:any` (y `xs:anyAttribute`) permiten usar cualquier elemento o atributo de los esquemas. En la imagen siguiente puede ver el elemento `any` en la vista Esquema de XMLSpy.

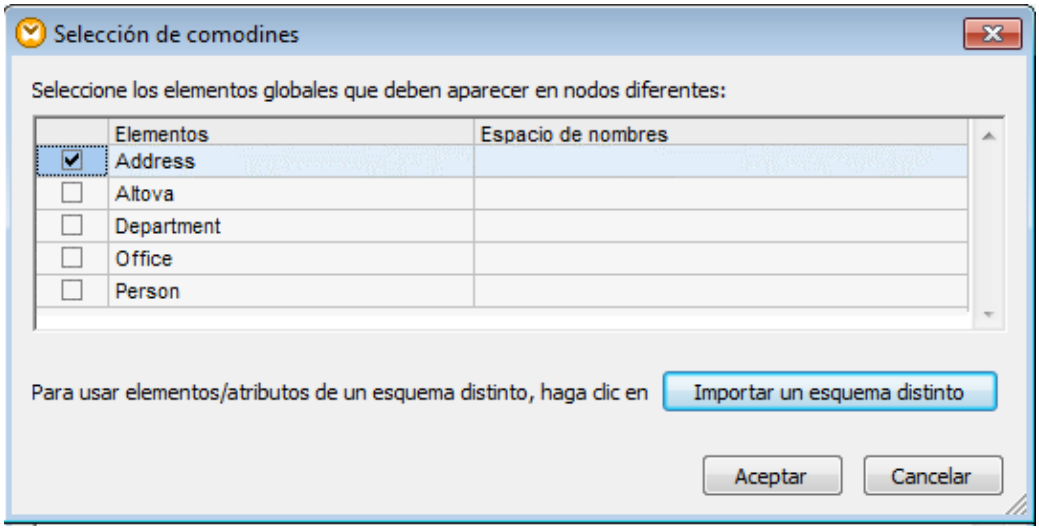


En MapForce los elementos `xs:any` (o `xs:anyAttribute`) aparecen acompañados del botón **Cambiar selección** .

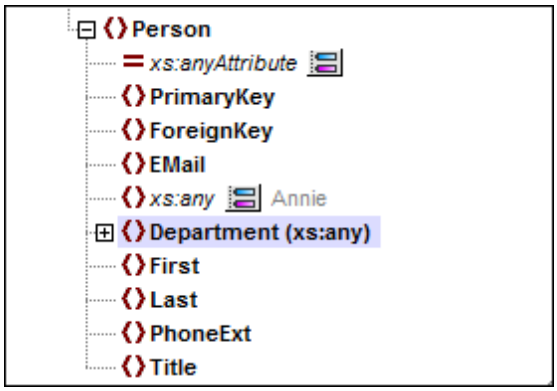


Cuando se hace clic en el botón **Cambiar selección** , aparece el cuadro de diálogo "Selección de comodines". En este cuadro de diálogo se enumeran los elementos y atributos

globales declarados en el esquema actual.



Si marca una de las casillas y hace clic en **Aceptar** para confirmar, MapForce inserta el elemento/atributo (y sus nodos secundarios) en el componente. Los elementos o atributos comodines se insertan inmediatamente después del nodo cuyo botón **Cambiar selección** accionó el usuario.



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)` que aparece anexado.

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

Comodines y nombres de nodo dinámicos

Por lo general, asignar datos a comodines o viceversa es adecuado si todos los atributos o elementos posibles que aparecen en la instancia XML están declarados por el esquema XML del componente (o si se pueden importar de esquemas externos). Sin embargo, en algunos casos

los elementos o atributos que aparecen en la instancia son demasiados. Por ejemplo, en esta instancia el número de elementos secundarios de <message> es aleatorio:

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

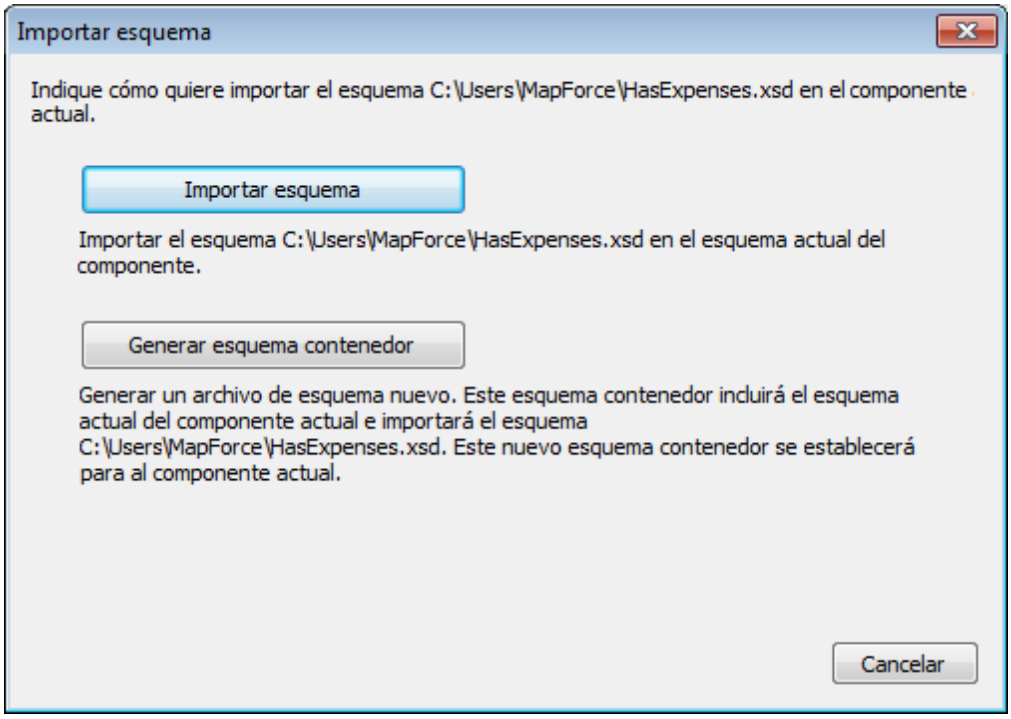
Para casos como este, en lugar de comodines, se recomienda un acceso dinámico a los nombres de nodo (*véase* *Asignar nombres de nodos*).

Agregar elementos de un esquema distinto como comodines

También puede 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** del cuadro de diálogo "Selección de comodines". Esto abre un cuadro de diálogo nuevo que ofrece dos opciones:

1. Importar un esquema o
2. generar un esquema contenedor.

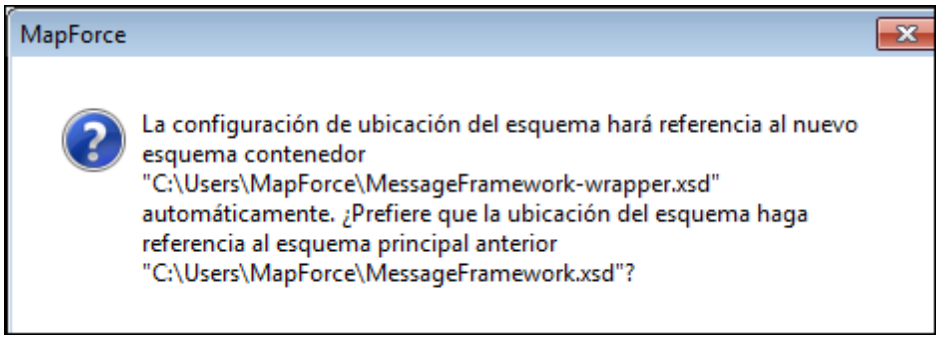
Por ejemplo, en la imagen siguiente puede ver lo que ocurre si intenta importar un esquema externo llamado **HasExpenses.xsd** en el esquema actual que está asignado a un componente.



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*) y no desde el disco, entonces no se podrá modificar. En este caso debe utilizarse la opción **Generar 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 **archivo-wrapper.xsd**. Una vez guardado, el esquema contenedor se asigna automáticamente y por defecto al componente y aparece este aviso:



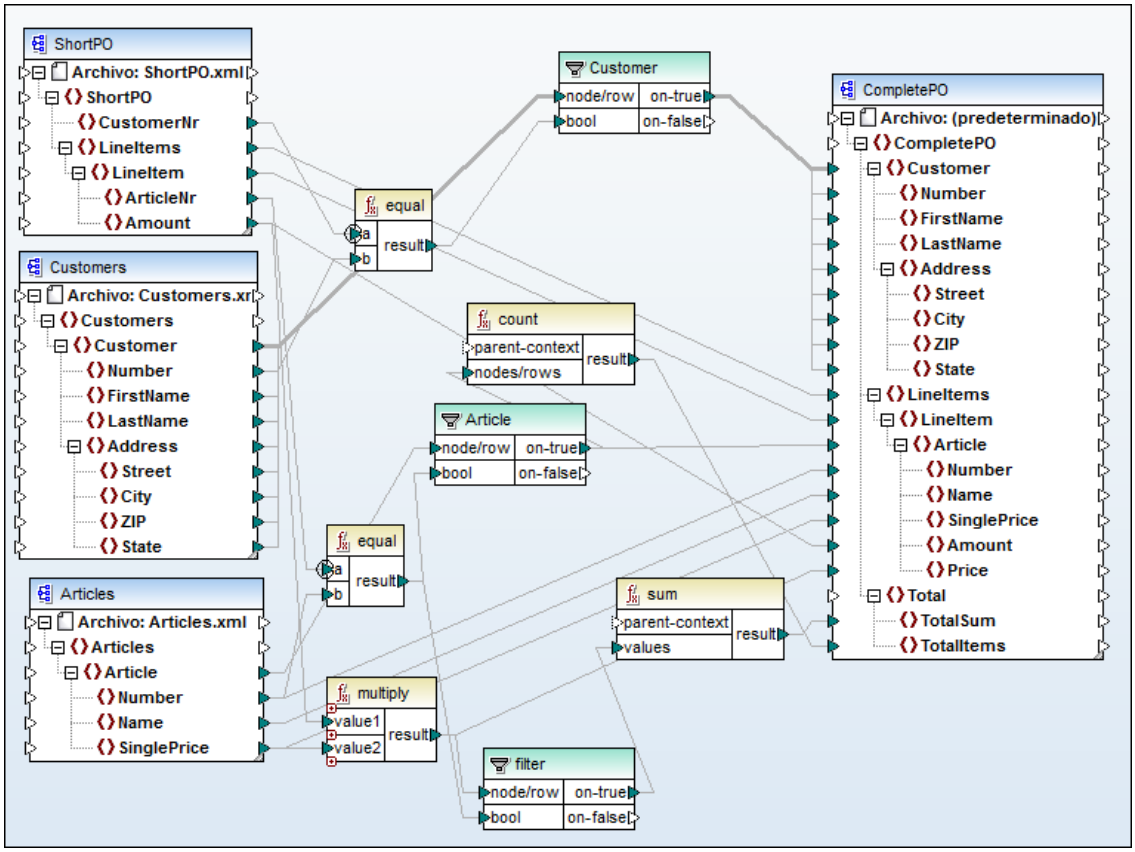
Haga clic en **Sí** para volver al esquema anterior y en **No** para que el esquema contenedor recién creado siga asignado al componente.

6.1.10 Combinar datos de varios esquemas distintos

MapForce permite combinar varios archivos en un solo archivo de destino.

El ejemplo de este apartado combina varios componentes de origen con esquemas distintos en un solo esquema de destino. En la sección Procesar varios archivos de entrada o salida simultáneamente se explica cómo combinar un número aleatorio de archivos usando el mismo esquema.

El archivo **CompletePO.mfd** de la carpeta `...\MapForceExamples` combina tres archivos XML en un archivo XML de orden de compra.



Observe que los datos de varios componentes de origen se combinan en un solo archivo XML de destino (CompletePO).

- **ShortPO** es un esquema que tiene asociado un archivo XML de instancia y contiene los datos sobre los artículos y el número de cliente (en este archivo solo hay un cliente con número de cliente 3).
- **Customers** es un esquema que tiene asociado un archivo XML de instancia y contiene el número de cada cliente e información sobre ellos como el nombre del cliente y su dirección.
- **Articles** es un esquema que tiene asociado un archivo XML de instancia y contiene datos sobre los artículos, como el precio, el nombre y el número de identificación.
- **CompletePO** es un esquema sin archivo de instancia asociado porque todos los datos vienen de los tres archivos XML de instancia. La estructura jerárquica de este archivo permite combinar todos los datos XML y darles salida.

Este archivo de esquema debe crearse en un editor XML como XMLSpy. No se genera con MapForce (aunque se podría crear si tuviera un archivo de instancia CompletePO.xml).

La estructura de CompletePO es una combinación de la estructura de los archivos XML de origen.

El componente de filtrado Customer sirve para buscar/filtrar los datos donde el número de cliente sea idéntico en los archivos XML ShortPO y Customers y después pasar los datos asociados al componente de destino CompletePO.

- El nodo **CustomerNr** de ShortPO se compara con el nodo **Number** de Customers con ayuda de la función "equal".

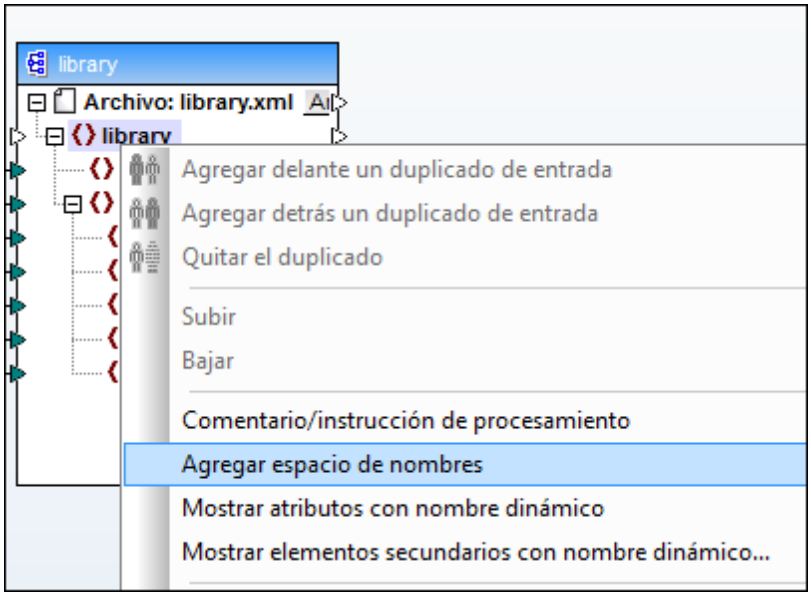
- Como ShortPO solo contiene un cliente (el número 3), solamente se puede pasar al componente de filtrado los datos disponibles para el cliente número 3.
- El parámetro **node/row** del componente de filtrado pasa los datos de **Customer** a "on-true" si el parámetro es `true`, es decir, cuando se encuentra el mismo número (en este caso el número 3).
- El resto de datos sobre el cliente y los artículos se pasan al esquema de destino a través de dos componentes de filtrado más.

6.1.11 Declarar espacios de nombres personalizados

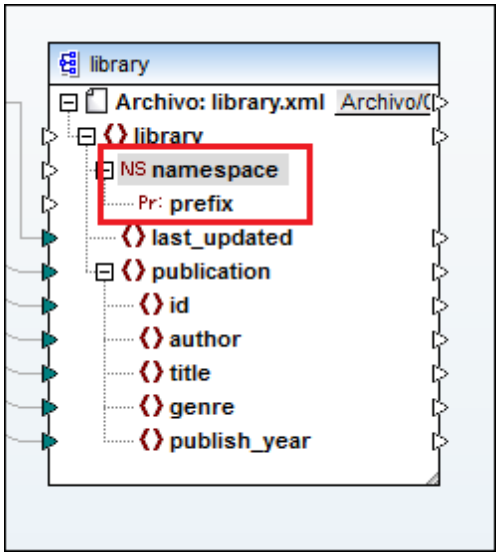
Cuando una asignación produce un resultado XML, MapForce deriva automáticamente el espacio de nombres (o conjunto de espacios de nombres) de cada elemento y atributo a partir del esquema asociado al componente de destino. Este es el comportamiento predeterminado de MapForce y el más apropiado en las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

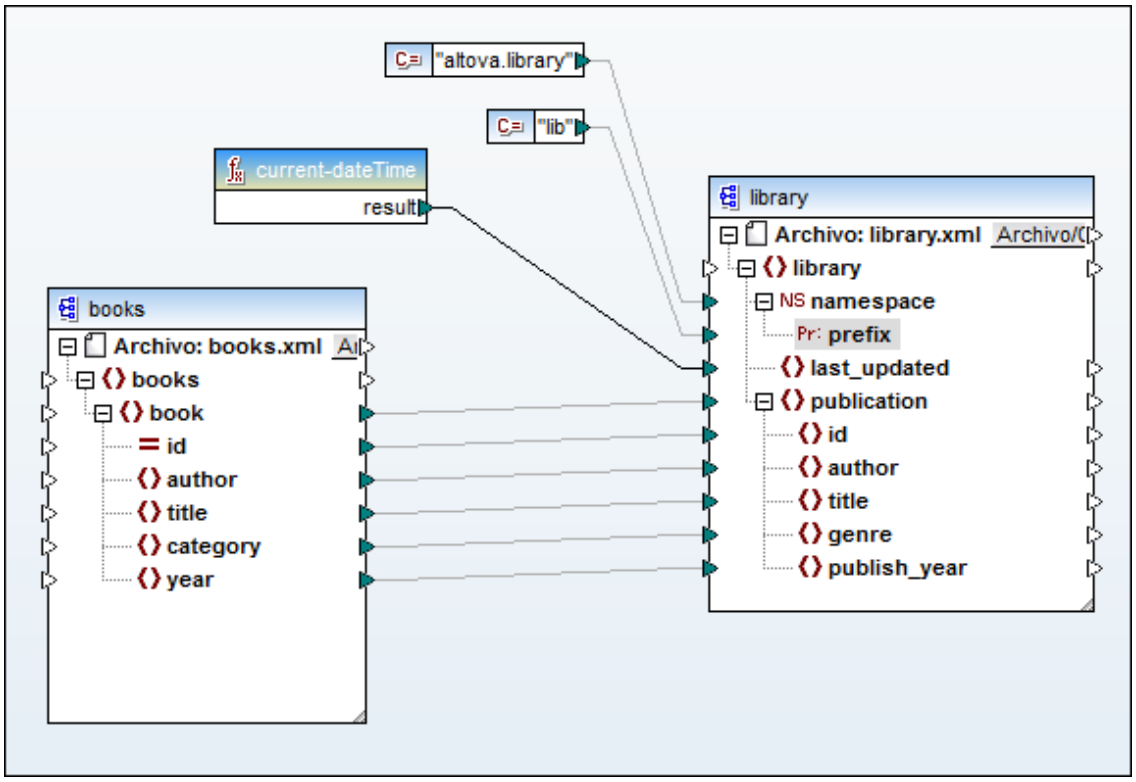
Para comprender mejor este funcionamiento abra la asignación **BooksToLibrary.mfd** de la carpeta `<Documentos>\Altova\MapForce2018\MapForceExamples\Tutorial\`. Haga clic con el botón derecho en el nodo `library` y seleccione Agregar **espacio de nombres** en el menú contextual.



Observe que ahora aparecen dos nodos nuevos bajo el nodo `library`: el nodo `namespace` y el nodo `prefix`.



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



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 (en este caso vienen dados por constantes). El resultado será:

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

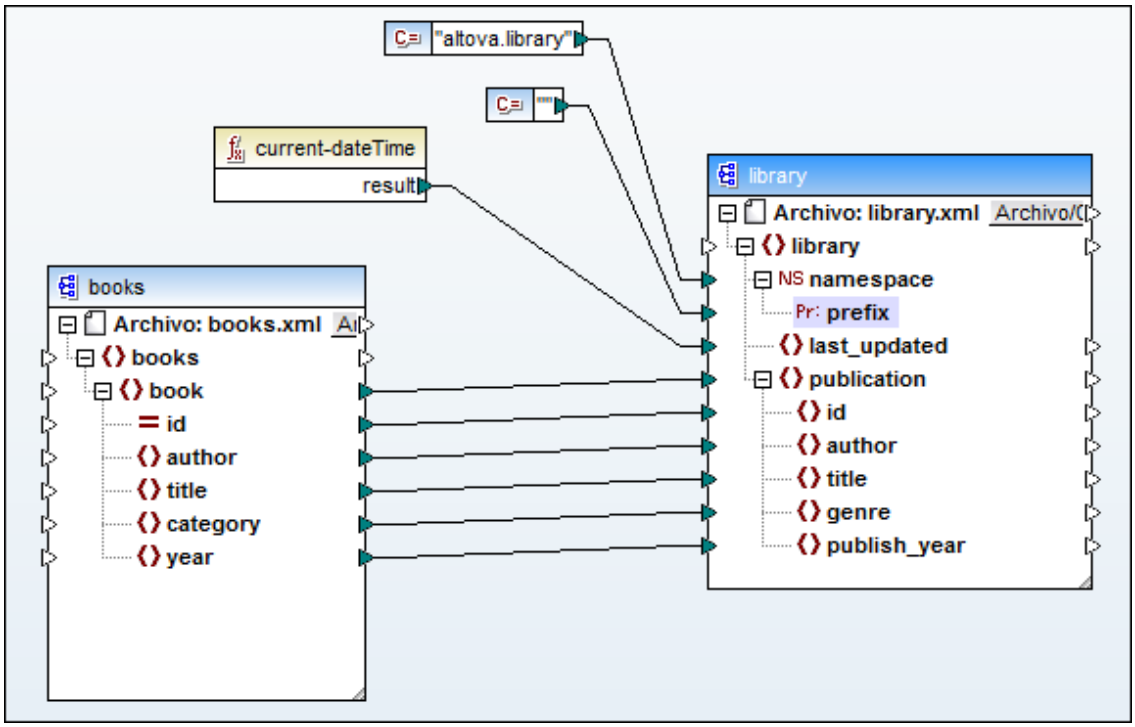
Nota: la declaración de espacios de nombres personalizados (y el comando **Agregar espacio de nombres**) solamente es relevante para componentes de destino XML y solamente para elementos. El comando **Agregar espacio de nombres** no está disponible en el caso de atributos ni nodos comodín. Tampoco está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

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.

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.

Es obligatorio conectar los conectores de entrada de `namespace` y de `prefix` (aunque estén conectados a valores vacíos).

Si desea declarar un espacio de nombres predeterminado (es decir, un espacio de nombres con el formato `xmlns="miEspacioPredeterminado"`), basta con asignar un valor de cadena vacío al nodo `prefix`. Por ejemplo, en la imagen siguiente, el valor de la segunda constante es una cadena vacía.



El resultado sería:

```
<?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*) o editar el esquema para que contenga un atributo `prod:id` para `<number>`.

6.2 HL7 versión 3

MapForce 2018 es automáticamente compatible con HL7 versión **3.x** porque este estándar está basado en XML.

En la página Componentes de MapForce del sitio web de Altova encontrará un instalador para los esquemas XML y los archivos de configuración HL7 V2.2 - V2.5.1. Durante la instalación seleccione la opción *Instalación personalizada* para instalar los componentes y esquemas XML HL7 V3 solamente.

Ubicación de los esquemas XML HL7 tras la instalación:

• MapForce de 32 bits en sistemas operativos de 32 bits • MapForce de 64 bits en sistemas operativos de 64 bits	`C:\Archivos de programa\Altova\Common2018\Schemas\hl7v3`
• MapForce de 32 bits en sistemas operativos de 64 bits	`C:\Archivos de programa(x86)\Altova\Common2018\Schemas\hl7v3`

En MapForce puede usar documentos HL7 como componentes de origen y destino. Estos datos también se pueden asignar a componentes de esquema XML.

Altova MapForce 2018 Basic Edition

Funciones

7 Funciones

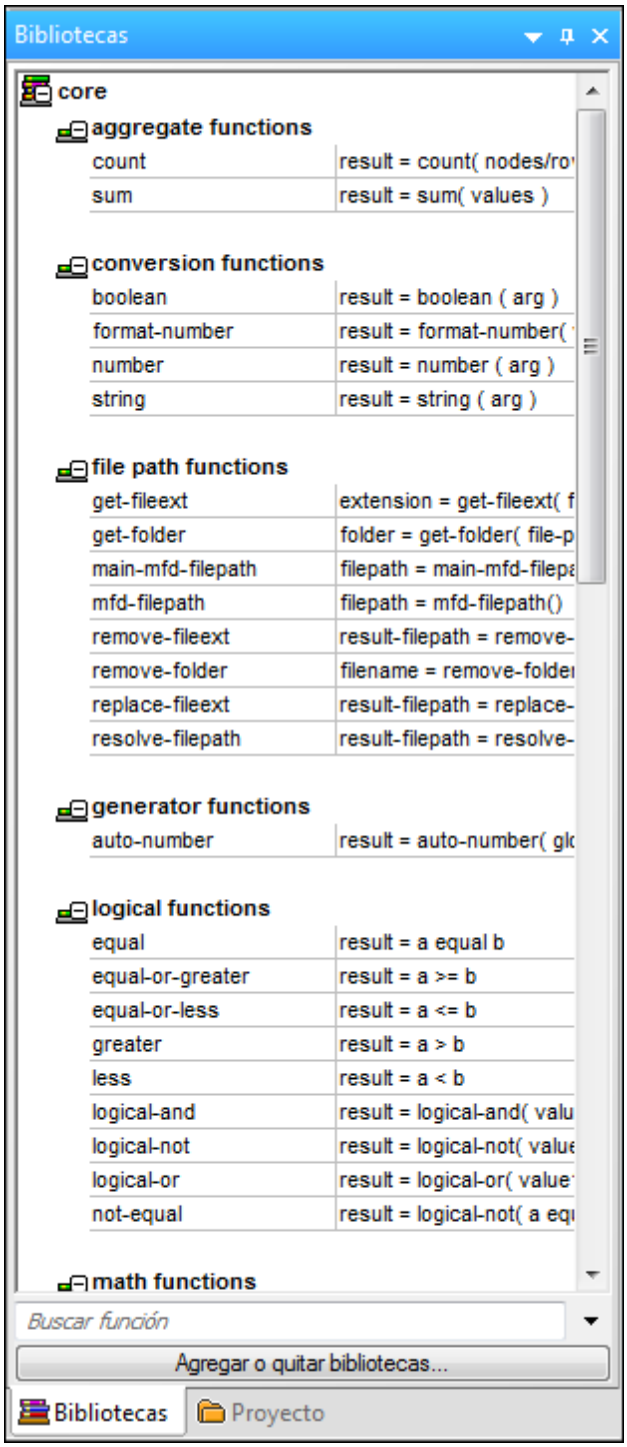
Las funciones suponen un potente mecanismo para transformar datos. En esta sección encontrará instrucciones para trabajar con funciones (tanto sin son funciones integradas de MapForce, como si son funciones definidas por el usuario o importadas de fuentes externas). A continuación puede ver un resumen de los temas que incluye esta sección:

Si le interesa...	Consulte este tema...
Aprender a trabajar con funciones en MapForce.	Trabajar con funciones
Crear sus propias funciones en MapForce.	Funciones definidas por el usuario
Agregar funciones XSLT personales a MapForce.	Importar funciones XSLT 1.0 o 2.0 personales
Ver todas las funciones integradas de MapForce o consultar la descripción de una función.	Referencia de la biblioteca de funciones

7.1 Trabajar con funciones

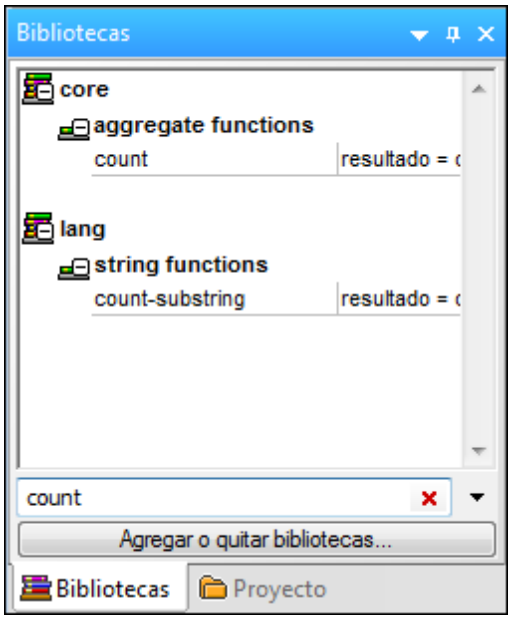
Para usar una función en una asignación de datos:

1. Seleccione el lenguaje de transformación (*véase* *Seleccionar el lenguaje de transformación*).
2. Haga clic en la función pertinente en la ventana Bibliotecas y arrástrela hasta el área de asignación.

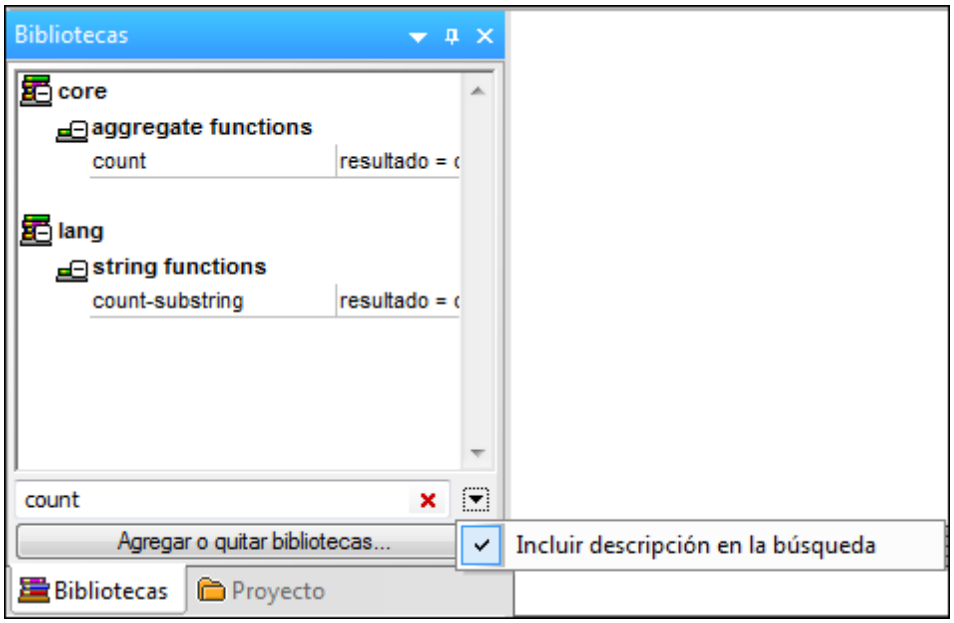


Para buscar una función en la ventana Bibliotecas:

1. Empiece a teclear el nombre de la función en el cuadro de texto situado en la parte inferior de la ventana Bibliotecas.



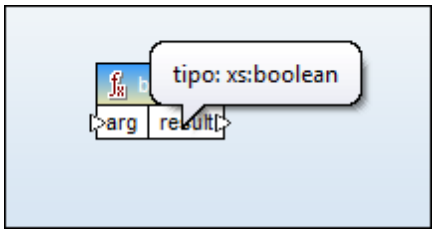
MapForce busca en el nombre de las funciones y en su descripción. Si desea excluir la descripción de las funciones de la búsqueda, haga clic en el icono en forma de flecha y deshabilite la opción **Incluir descripción en la búsqueda**.



Para cancelar la búsqueda pulse la tecla **Esc** o haga clic en el icono ×.

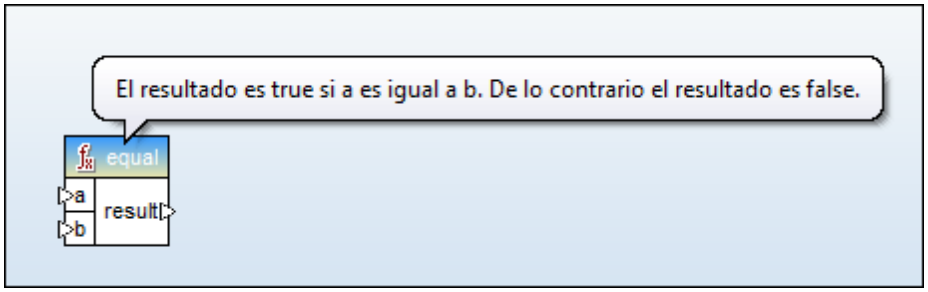
Para ver el tipo de datos de un argumento de entrada/salida de una función:

1. Compruebe que está activado el botón **Mostrar información rápida** de la barra de herramientas.
2. Pase el puntero por encima del argumento de la función.



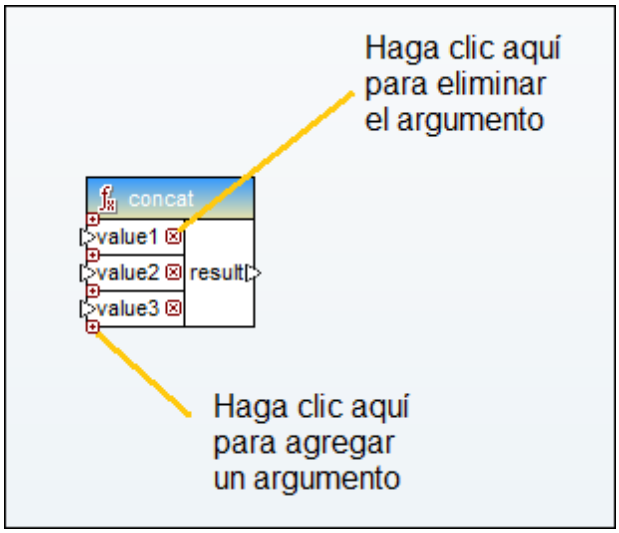
Para ver la descripción de una función:

1. Compruebe que está activado el botón **Mostrar información rápida** de la barra de herramientas.
2. Pase el puntero por encima de la función (este mecanismo funciona tanto en la ventana Bibliotecas como en el área de asignación)



Para agregar o eliminar argumentos de las funciones (si procede):

- Haga clic en el icono **Agregar parámetro** (⊡) o **Eliminar parámetro** (⊠) del parámetro que desea agregar/eliminar respectivamente.



Si arrastra y coloca una conexión en el icono ⊡, MapForce añade automáticamente el

parámetro y lo conecta.

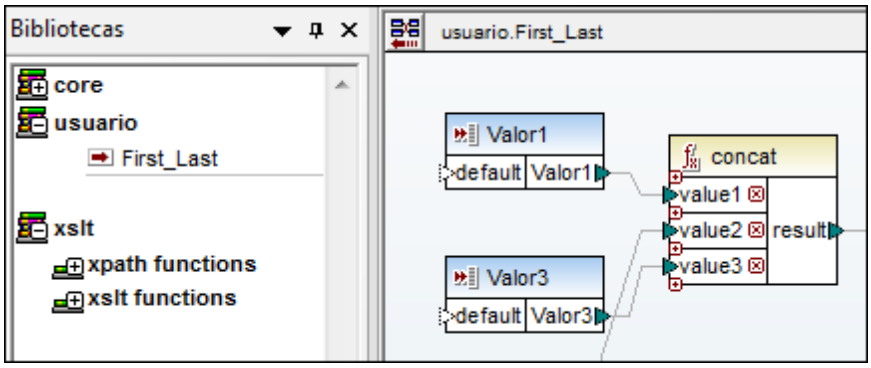
Para buscar todas las instancias de una función dentro de la asignación de datos activa:

- Haga clic con el botón derecho en el nombre de la función en la ventana Bibliotecas y elija **Buscar todas las llamadas** en el menú contextual. El resultado de la búsqueda aparece en la ventana Mensajes.

7.2 Funciones definidas por el usuario

En MapForce puede crear funciones definidas por el usuario de forma visual, con las mismas operaciones que se usan en el panel *Asignación*.

Una vez definidas, las funciones están disponibles en la ventana Bibliotecas (*ver la función* ***First_Last*** *de la imagen*) y se utilizan igual que las demás funciones de MapForce. Las funciones definidas por el usuario permiten organizar la asignación en varios bloques que se pueden volver a utilizar en la misma asignación o en otras asignaciones.



Las funciones definidas por el usuario se almacenan en el archivo ***.mfd**, junto con la asignación principal.

Una función definida por el usuario utiliza **componentes** de **entrada** y **salida** para pasar la información desde la asignación principal (o desde otra función definida por el usuario) hasta la función definida por el usuario y otra vez a la asignación principal.

Las funciones definidas por el usuario pueden tener componentes de origen locales (es decir, situados dentro de la función propiamente dicha), como por ejemplo esquemas XML, lo cual es práctico a la hora de implementar funciones de búsqueda.

Las funciones definidas por el usuario pueden tener un número ilimitado de entradas y salidas. Éstas pueden ser valores simples, o nodos XML.

Las funciones definidas por el usuario permiten:

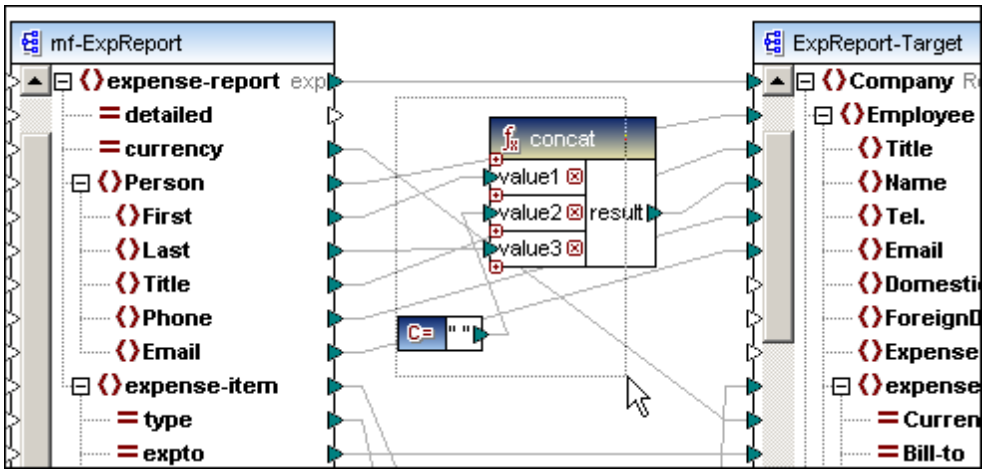
- reunir varias funciones de procesamiento en un solo componente (p. ej. para dar formato a un campo concreto o para buscar un valor)
- reutilizar estos componentes un número ilimitado de veces
- importar funciones definidas por el usuario en otras asignaciones (cargando el archivo de asignación como biblioteca)
- usar funciones inline para descomponer una asignación compleja en varias partes más pequeñas para editarlas por separado
- asignar esquemas recursivos creando funciones definidas por el usuario recursivas

Las funciones definidas por el usuario se pueden crear desde cero o a partir de otras funciones ya disponibles en el panel *Asignación*.

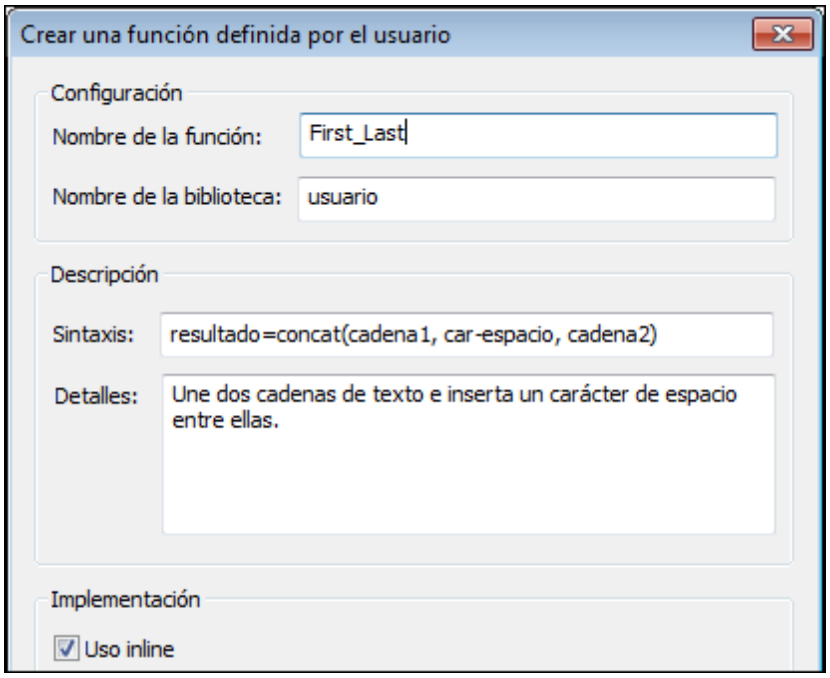
El ejemplo que aparece a continuación se basa en el archivo **Tut-ExpReport.mfd** de la carpeta ...\MapForceExamples\Tutorial\.

Crear una función definida por el usuario a partir de componentes de la asignación

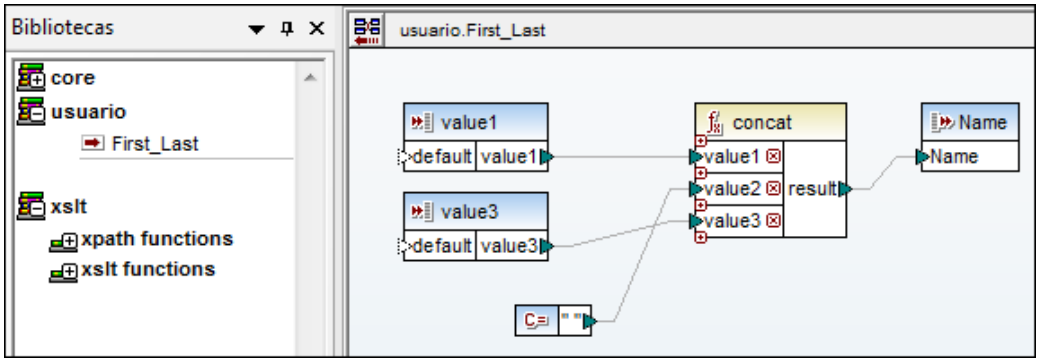
1. Arrastre el puntero por el área de asignación para seleccionar la constante y la función **concat** (también puede hacer clic en estos componentes mientras pulsa la tecla **Ctrl**).



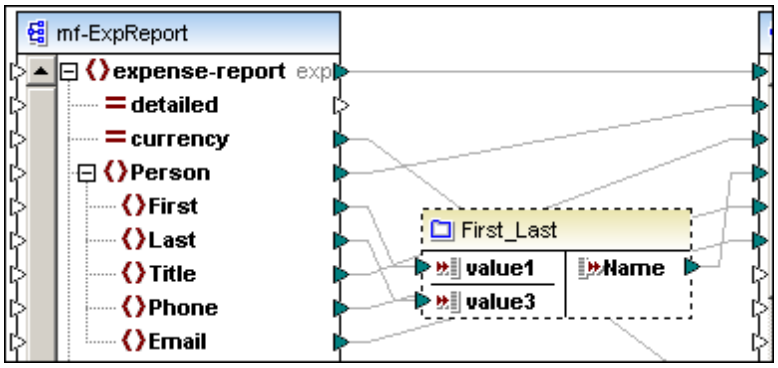
2. Seleccione la opción de menú **Función | Crear una función definida por el usuario a partir de la selección**.
3. Aparece un cuadro de diálogo donde puede escribir el nombre de la nueva función definida por el usuario (es decir, **First_Last**).
   Nota: los caracteres válidos para el nombre de la función son los caracteres alfanuméricos (a-z, A-Z, 0-9), el carácter de subrayado (_), el guión (-) y los dos puntos (:).
4. En los campos *Sintaxis* y *Detalles* puede incluir más información sobre la nueva función. Cuando termine haga clic en **Aceptar**. El texto introducido en el campo *Detalles* aparece al pasar el puntero encima de la función. El nombre de biblioteca predeterminado para las funciones de usuario nuevas es **user**, pero puede crear un nombre de biblioteca nuevo en el campo *Nombre de la biblioteca*.



Los elementos que componen la función aparecen en un panel nuevo cuyo título es el nombre de la función. La biblioteca **usuario** aparece en el la ventana Bibliotecas y debajo el nombre de la función recién creada **`First_Last`**.



Haga clic en el botón **Volver**  para volver a la ventana de asignación principal. Observe cómo los componentes se reunieron en una sola función llamada **`First_Last`** y sus parámetros de entrada y salida se conectaron automáticamente.



Observe que las funciones definidas por el usuario tienen un contorno de guiones. Para más información consulte el apartado Funciones definidas por el usuario estándar e inline.

Para usar la función en la asignación actual, haga clic en el nombre de la función en la ventana Bibliotecas y arrástrela hasta el panel *Asignación*. Para aprender a usarla en otras asignaciones consulte las el apartado instrucciones que aparecen más abajo

Abrir funciones definidas por el usuario

Las funciones definidas por el usuario se pueden abrir de dos maneras distintas:

- haciendo doble clic en la barra de título de la función definida por el usuario en el panel *Asignación* o
- haciendo doble clic en la función definida por el usuario en la ventana Bibliotecas.

Acto seguido se abre un panel en el que aparecen los componentes que forman la función. Para volver a la asignación principal haga clic en el botón **Volver**  . Al hacer doble clic en una función definida por el usuario de un archivo *.mfd diferente (en el panel principal de asignación) se abre ese archivo *.mfd en otra pestaña.

Navegar por las funciones definidas por el usuario

Mientras se navega por varias pestañas (o paneles de funciones definidas por el usuario), MapForce genera automáticamente un historial para que pueda recorrer las pestañas/los paneles, haciendo clic en los iconos **Adelante/Atrás**. El historial es válido para toda la sesión, es decir puede cambiar de un archivo *.mfd a otro mientras retrocede/avanza por las pestañas.

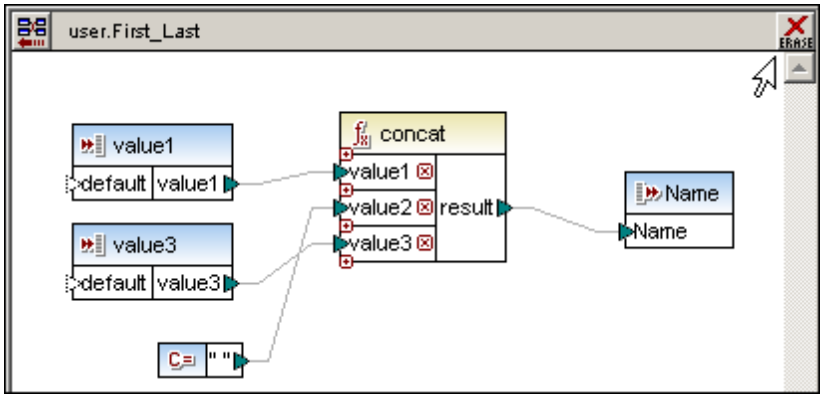
 Pulse el botón **Volver** para volver a la pestaña de asignación principal.

 Pulse el botón **Atrás** para retroceder en el historial.

 Pulse el botón **Adelante** para avanzar en el historial.

Eliminar funciones definidas por el usuario en la biblioteca

1. Haga doble clic en la función definida por el usuario en la ventana Bibliotecas.
2. Haga clic en el botón **Borrar** situado en la esquina superior derecha del panel.



Reutilizar (importar) funciones definidas por el usuario

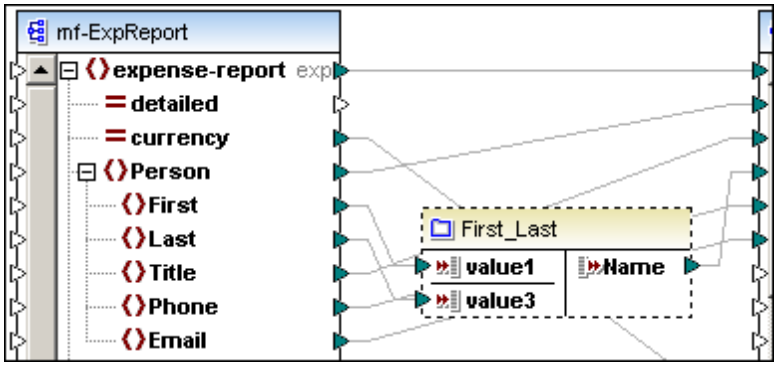
Las funciones definidas por el usuario en una asignación se pueden importar en otra asignación diferente:

1. Haga clic en el botón **Agregar o quitar bibliotecas** en la parte inferior de la ventana Bibliotecas.
2. En el cuadro de diálogo que aparece haga clic en el botón **Agregar** y seleccione el archivo *.mfd que contiene las funciones definidas por el usuario que desea importar. La función definida por el usuario aparece ahora en la ventana Bibliotecas. Si creó la función definida por el usuario con el nombre de biblioteca predeterminado, la biblioteca importada se llamará **user**. Si no se llama así, busque el nombre de biblioteca que especificó cuando creó la función definida por el usuario.
2. Arrastre la función que acaba de importar desde la ventana Bibliotecas hasta el área de asignación.

> Si se usa el mismo nombre de biblioteca en varios archivos *.mfd, todas las funciones de todas las fuentes disponibles aparecerán bajo el mismo nombre de biblioteca (en la ventana Bibliotecas). Sin embargo, solamente se pueden editar con doble clic las funciones del documento que esté activo en ese momento.

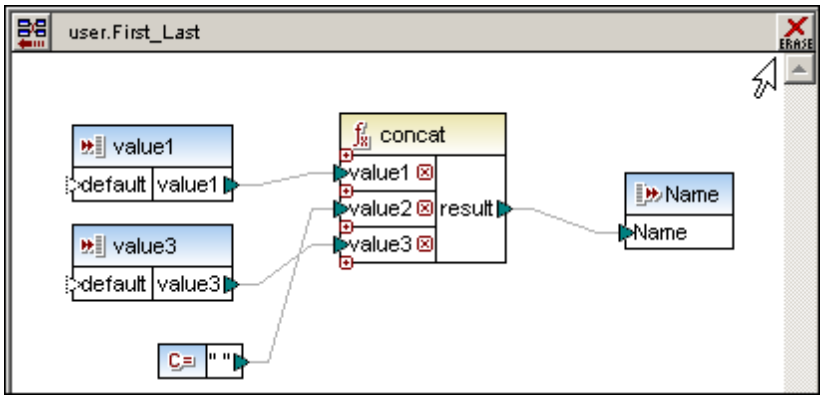
No olvide que, si realiza cambios en una función importada, cuando guarde el archivo ***.mfd** los cambios se aplicarán también a las asignaciones desde las que se importó la función.

Orden de los parámetros en funciones definidas por el usuario



El orden de los parámetros de las funciones definidas por el usuario depende de varios aspectos:

- los parámetros de entrada y salida se ordenan según su posición, de arriba a abajo (empezando por la esquina superior izquierda).
- si dos parámetros tienen la misma posición vertical, tiene prioridad el que esté situado más a la izquierda.
- en el improbable caso que dos parámetros tengan exactamente la misma posición, el identificador interno del componente se usa automáticamente.



Notas:

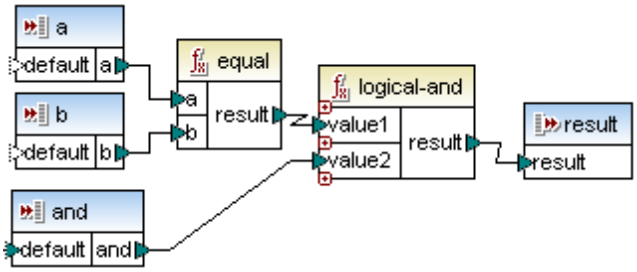
- la posición del componente y las acciones de ajuste de tamaño no se pueden deshacer.
- los componentes de entrada y salida recién añadidos se crean bajo la entrada y salida creada inmediatamente antes.
- puede mezclar parámetros complejos y simples. El orden de los parámetros se deduce de la posición del componente.

7.2.1 Parámetros de las funciones

Los **parámetros** de las funciones se representan dentro de las funciones definidas por el usuario en forma de **componentes** de **entrada** y **componentes** de **salida**.

- Componentes/parámetros de entrada: **a**, **b**, **and**

- Componentes/parámetros de salida: **result**

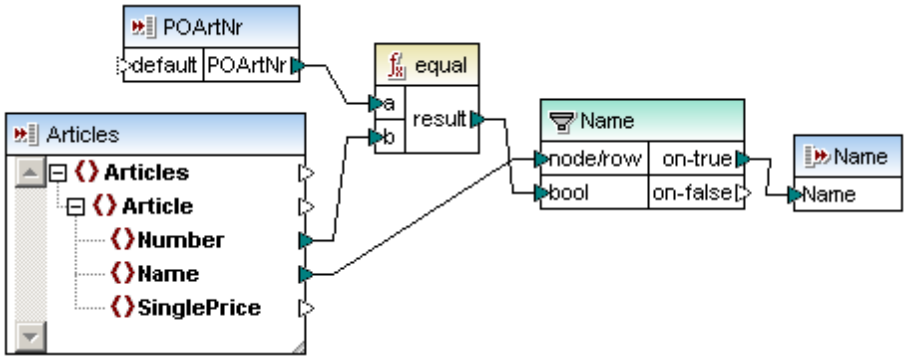


Los parámetros de entrada se utilizan para pasar datos de la asignación principal a la función definida por el usuario. Los parámetros de salida se utilizan para devolver los datos a la asignación principal. Recuerde que a las funciones definidas por el usuario se les puede llamar desde otras funciones definidas por el usuario.

Parámetros simples y complejos

Los parámetros de entrada y salida de las funciones definidas por el usuario pueden ser de varios tipos:

- Valores simples (p. ej. `string` o `integer`)
- Árboles de nodos complejos (p. ej. un elemento XML con atributos y nodos secundarios)



El parámetro de entrada **`POArtNr`** es un valor **simple** de tipo `string`

El parámetro de entrada **`Articles`** es un árbol de nodos XML **complejo** (un documento XML)

El parámetro de salida **`Name`** es un valor simple de tipo `string`

Nota: las funciones definidas por el usuario de las dos imágenes anteriores están disponibles en el archivo de ejemplo **`PersonListByBranchOffice.mfd`** , guardado en la carpeta `...\MapForceExamples`.

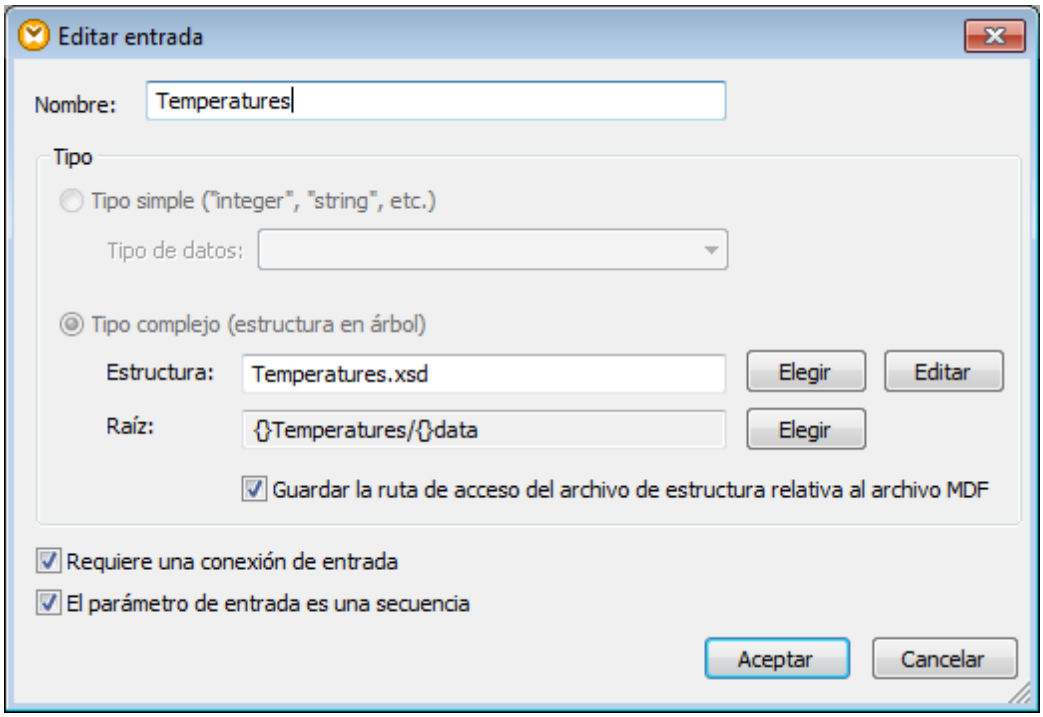
Secuencias

Las secuencias son datos compuestos por un rango (o secuencia) de valores. Los **parámetros** simples y complejos de las funciones definidas por el usuario (sus entradas y salidas) se pueden definir como secuencias en el cuadro de diálogo de propiedades del componente.

En las funciones de agregado (p. ej. **`min`**, **`max`**, **`avg`**, etc.) puede usar este tipo de entradas para

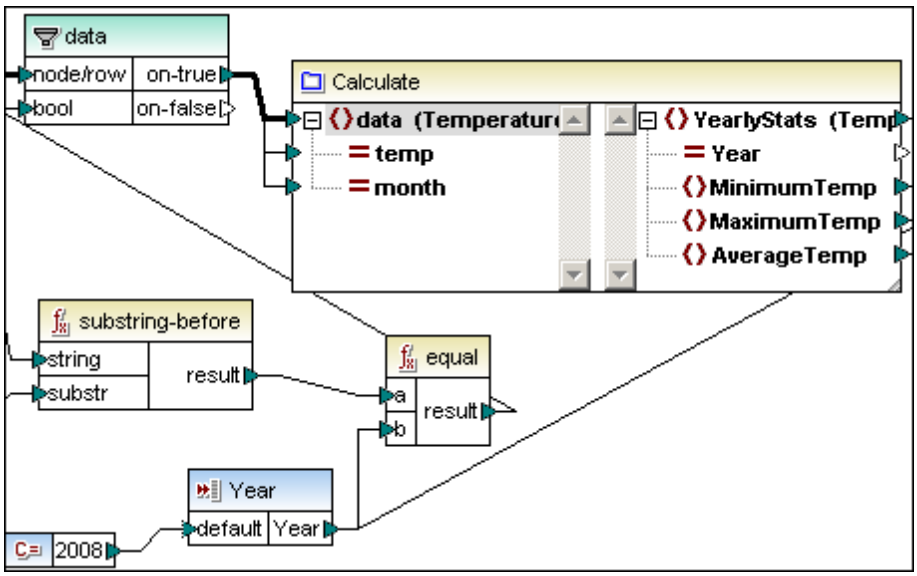
suministrar un valor determinado de la secuencia de entrada.

Si activa la casilla *El parámetro de entrada es una secuencia,* el componente trata la entrada como si fuera una secuencia de valores. Si desactiva la casilla, la entrada se trata como si fuera un valor único.



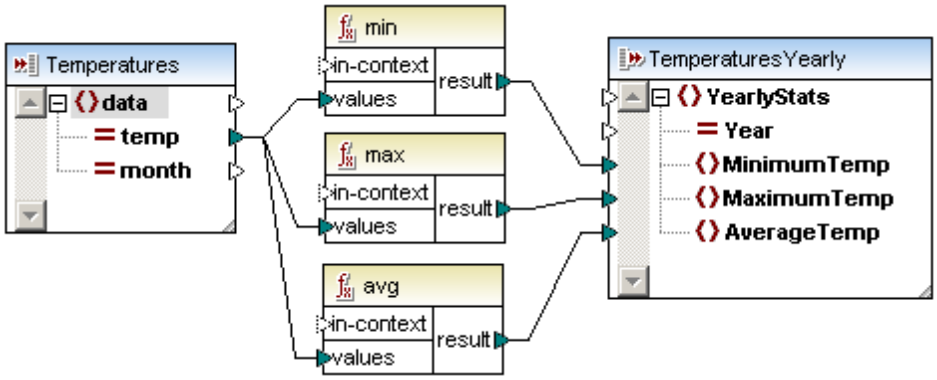
Este tipo de datos de entrada (de secuencia o no secuencia) determina **con qué frecuencia** se llama a la función.

- Cuando la función definida por el usuario está conectada a un parámetro **secuencial**, se le llama **una sola vez** y toda la secuencia se pasa a la función definida por el usuario.



La imagen anterior muestra la función definida por el usuario **`Calculate`**, del archivo de

asignación `InputIsSequence.mfd` (disponible en la carpeta `...\MapForceExamples`). El componente de entrada **`Temperatures`** (*imagen siguiente*) se definió como secuencia.



- Cuando la función definida por el usuario está conectada a un parámetro **no secuencial**, se le llama **una vez por cada elemento** de la secuencia.

Nota: la configuración secuencial de los parámetros de entrada/salida se ignora cuando la función definida por el usuario es de tipo inline.

Es importante tener en cuenta que, si conecta una secuencia vacía a un parámetro no secuencial, a la función no se le llama en absoluto.

Esto puede ocurrir si la estructura de origen tiene elementos **opcionales** o cuando una condición de un filtro no devuelve elementos. Para evitar que esto ocurra utilice la función substitute-missing antes de la entrada de la función (evitando que la secuencia esté vacía) o defina la entrada como secuencia y defina el procesamiento de secuencias vacías dentro de la función.

Cuando una función pasa una secuencia de varios valores a su componente de salida y el componete de salida no se definió como secuencia, solamente se usa el primer resultado cuando se llama a la función.

7.2.2 Funciones definidas por el usuario estándar e inline

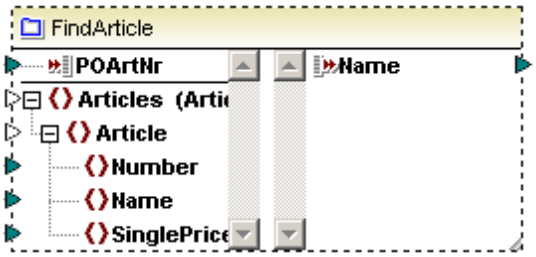
Las funciones inline son diferentes de las demás funciones por la manera en que se implementan cuando se genera código.

- El código para las funciones de tipo **inline** se inserta en **todas las posiciones** donde se llama a las funciones definidas por el usuario o donde se utilizan
- El código de una función **normal** se implementa como una llamada a función

Por tanto, es como si las funciones inline fueran sustituidas por su implementación. Por este motivo son ideales para descomponer asignaciones complejas en varias partes más pequeñas.

Nota: el uso de funciones **inline** incrementa significativamente la cantidad de código de programa generado porque el código de la función definida por el usuario se inserta en todas las posiciones donde se llama a la función o donde se utiliza.

Las **funciones definidas por el usuario** de tipo **INLINE** tienen un contorno de guiones:



Las funciones definidas por el usuario inline **admiten:**

- **varios componentes de salida**

Las funciones definidas por el usuario inline **no admiten:**

- parámetros configurados como contexto prioritario
- llamadas recursivas a una función definida por el usuario inline

Las **funciones definidas por el usuario** de tipo **ESTÁNDAR** (es decir, no inline) tienen un contorno sólido:

Las funciones definidas por el usuario de tipo estándar (no inline) **admiten:**

- un solo componente de salida
- **llamadas recursivas** (aquellas en las que se debe dar la condición de salida, p. ej. usar una condición If-Else en la que una rama o valor sale de la recursión)
- parámetros configurados como contexto prioritario

Nota: aunque las funciones estándar no admiten varios componentes de salida, estos componentes se pueden crear en este tipo de función. Sin embargo, cuando intente generar código o ver una vista previa del resultado de la asignación, aparecerá un mensaje de error en la ventana Mensajes.

Si no usa una recursión en su función, puede cambiar el tipo de función a inline.

Las funciones definidas por el usuario de tipo estándar (no inline) **no admiten:**

- la conexión directa de filtros a componentes de **entrada** simples no secuenciales
- el uso de funciones de secuencia o agregado en componentes de entrada simples (como **`exists`**, **`substitute-missing`**, **`sum`**, **`group-by`**, etc.)

Nota sobre generación de código

La implementación de una función definida por el usuario de tipo estándar se genera una sola vez como función o plantilla XSLT a la que se puede llamar. Cada componente de la función definida por el usuario genera código para una **llamada a función**, en la que las entradas se pasan como parámetros y la salida es el valor devuelto de la función (componente).

En tiempo de ejecución se evalúan primero todos los valores del parámetro de entrada y después

se llama a la función cada vez que aparecen los datos de entrada. Para más información consulte el apartado Parámetros de las funciones.

Para cambiar de tipo de función definida por el usuario:

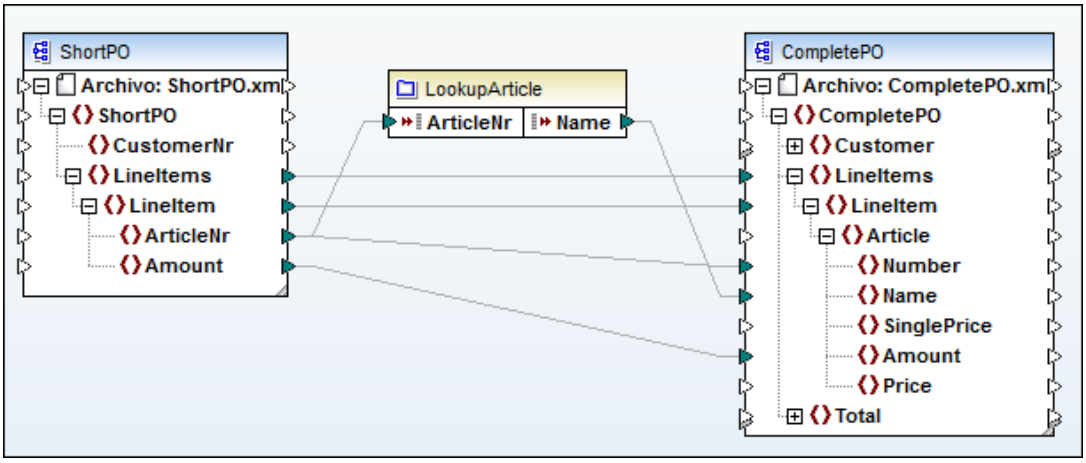
1. Haga doble clic en la función definida por el usuario para ver sus componentes.
2. Seleccione la opción de menú **Función | Configuración de la función** y active/ desactive la casilla *Uso inline*.

Funciones definidas por el usuario y conexiones de copia total
Cuando cree conexiones de copia total entre un esquema y un parámetro complejo de una función definida por el usuario, ambos componentes deben estar basados en el mismo esquema. Sin embargo, no es necesario que tengan el mismo elemento raíz. Para ver un ejemplo consulte el apartado Definir componentes de salida complejos.

7.2.3 Crear una función de búsqueda simple

Este apartado utiliza el archivo de ejemplo `lookup-standard.mfd`, disponible en la carpeta **...\MapForceExamples**.

Objetivo:
Crear una función de búsqueda genérica que suministre los datos de **`Articles/Number`** del archivo XML **Articles** para compararlos con los datos de **`Article`** de otro archivo XML, el archivo **ShortPO**.

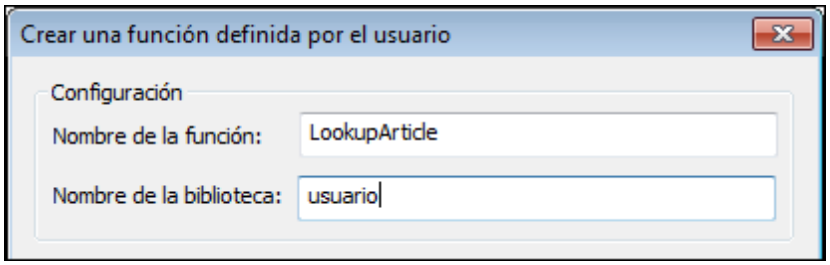


Instrucciones generales:

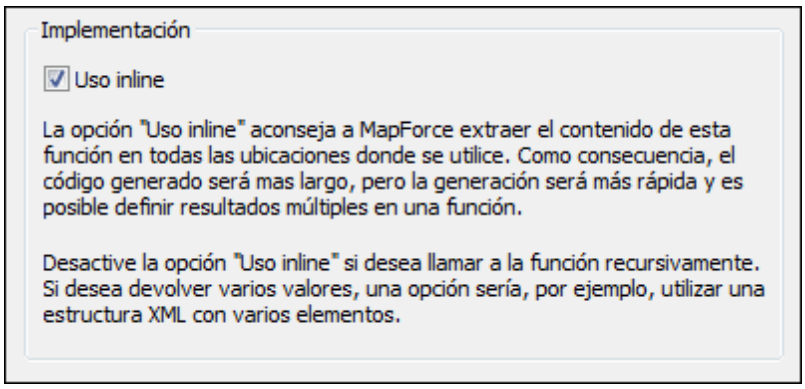
- Inserte el esquema `ShortPO.xsd` y asigne el archivo `ShortPO.xml` como archivo XML de entrada.
- Inserte el esquema `CompletePO.xsd` y seleccione el elemento raíz **`CompletePO`**.
- Inserte una función definida por el usuario nueva siguiendo los pasos descritos a continuación.

Para crear una función definida por el usuario:

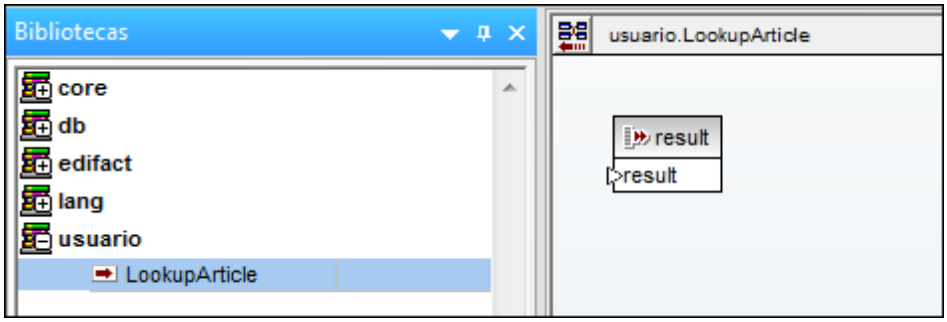
1. Seleccione la opción de menú **Función | Crear función definida por el usuario**.
2. En el cuadro de diálogo que aparece escriba el nombre de la función, p. ej. **`LookupArticle`**, y la biblioteca en la que desea almacenarla (p. ej. **`usuario`**).



3. Desactive la casilla *Uso inline* y haga clic en **Aceptar** para confirmar.



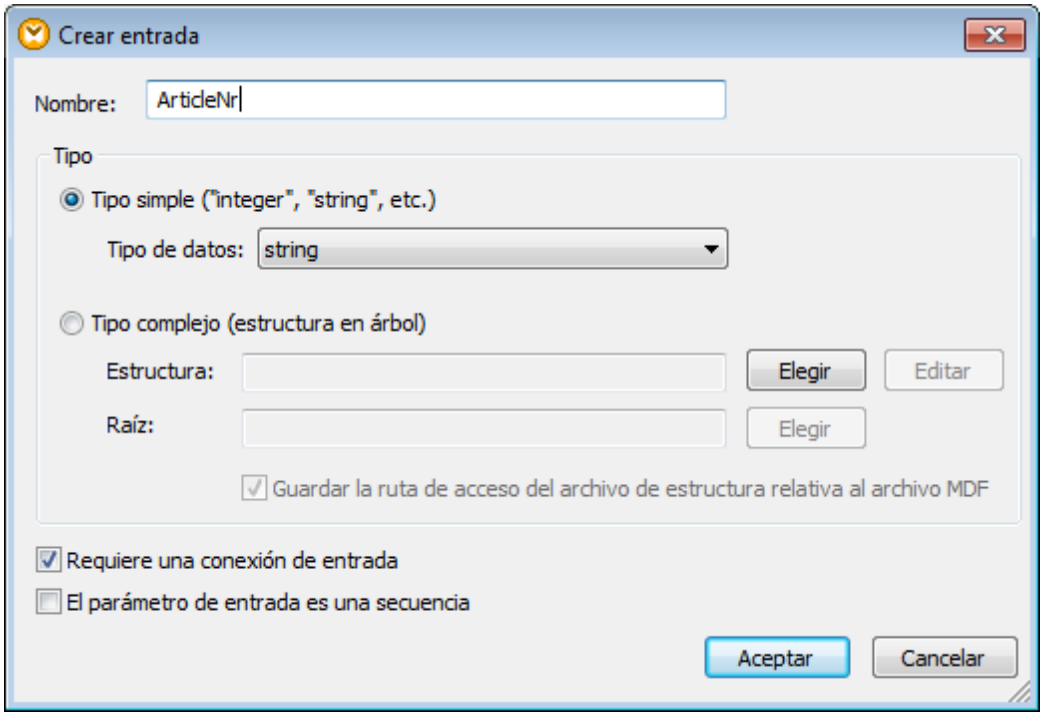
Aparece un panel con un solo elemento: un parámetro de salida llamado **`result`**.



Este panel es el área de trabajo donde podrá definir la función definida por el usuario.

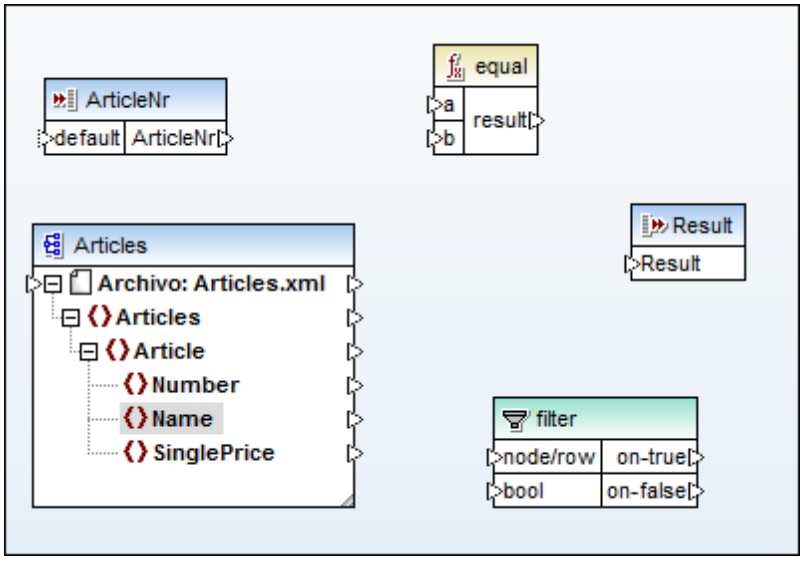
En la ventana **Bibliotecas** se creó una biblioteca nueva llamada **`usuario`** y bajo ella aparece la función **`LookupArticle`**.

4. Haga clic en el icono **Insertar esquema o archivo XML** para insertar el esquema **`Articles`** y seleccionar el archivo XML **`Articles`** como origen de datos.
5. Haga clic en el icono **Insertar componente de entrada** para insertar un componente de entrada.
6. Escriba el nombre del parámetro de entrada (**`ArticleNr`**) y haga clic en **Aceptar**.



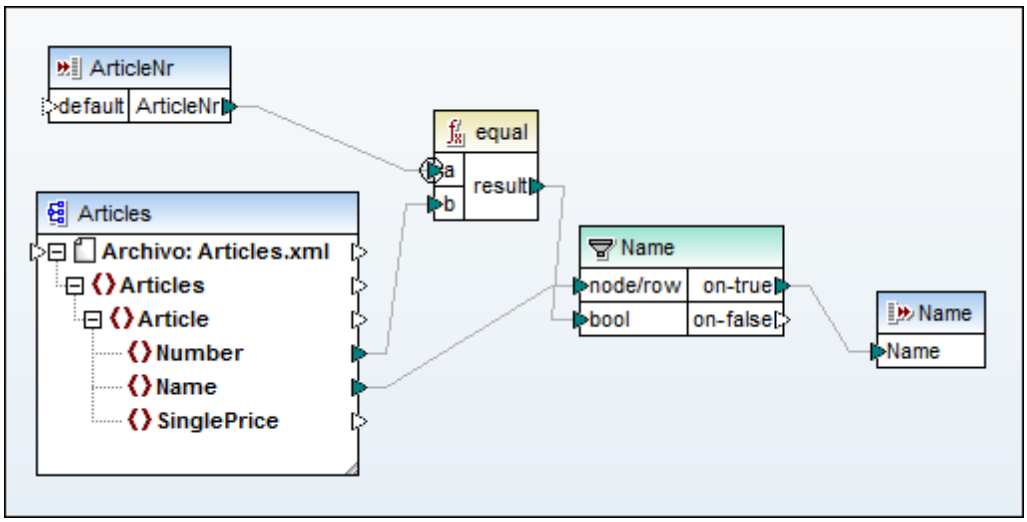
Este componente funciona como entrada de datos para la función definida por el usuario y constituye el icono de entrada de la función definida por el usuario.

7. Inserte la función **equal** de la biblioteca **core | logical**.
8. Inserte un componente de filtrado haciendo clic en el icono **Insertar filtro** de la barra de herramientas.



Ahora cree las asignaciones que aparecen en la imagen siguiente entre los componentes de la función definida por el usuario.

9. Tras crear las asignaciones haga clic con el botón derecho en el parámetro **a** de la función **equal** y seleccione **Contexto prioritario** en el menú contextual.
10. Haga **doble clic** en la salida de la función y escriba el nombre **Name** para el parámetro de salida.



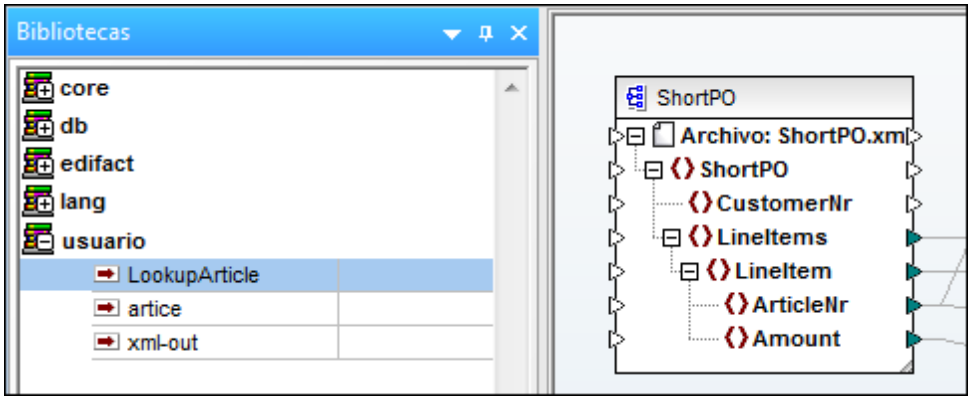
La función ya está definida.

Nota: al hacer doble clic en las entradas y salidas de la función aparece un cuadro de diálogo donde puede: (i) cambiar el nombre y el tipo de datos del parámetro, (ii) definir si la función debe tener un icono de entrada (*Requiere una conexión de entrada*) y (iii) definir el parámetro como secuencia.

La función definida por el usuario recién creada:

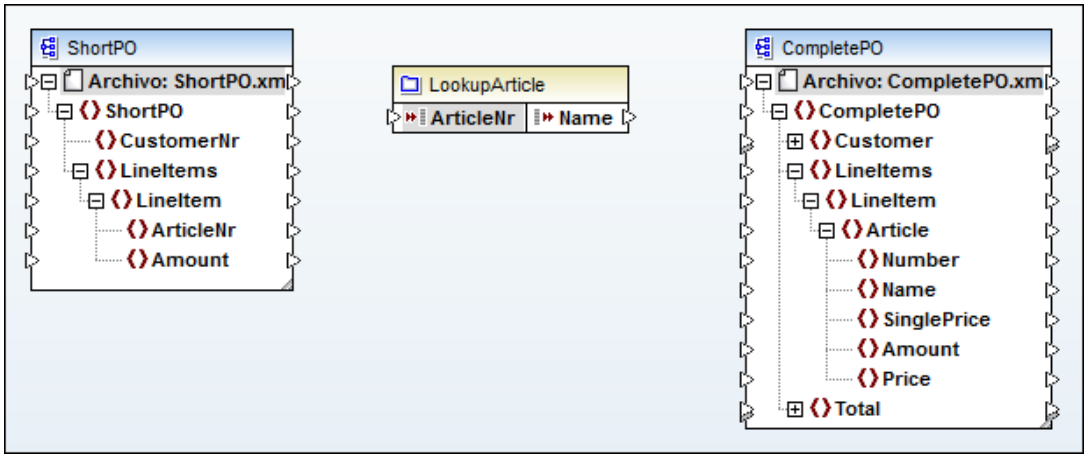
- tiene una **entrada** (**`ArticleNr`**) que recibirá los datos del archivo XML **ShortPO** en la asignación principal.
- **compara** los datos de **`ArticleNr`** del archivo XML **ShortPO** con los datos de **`Article/Number`** del archivo XML **Articles** (que se insertó en la función con dicho objetivo).
- utiliza un **filtro** para enviar los registros de **`Article/Name`** al componente de salida si el resultado de la comparación es **`true`**.
- tiene una **salida** (**`Name`**) que enviará los registros de **`Article/Name`** al archivo XML de destino **CompletePO**.

11. Haga clic en el icono **Volver** para volver a la asignación principal.
La función definida por el usuario **`LookupArticle`** ahora está disponible en la biblioteca **usuario**.

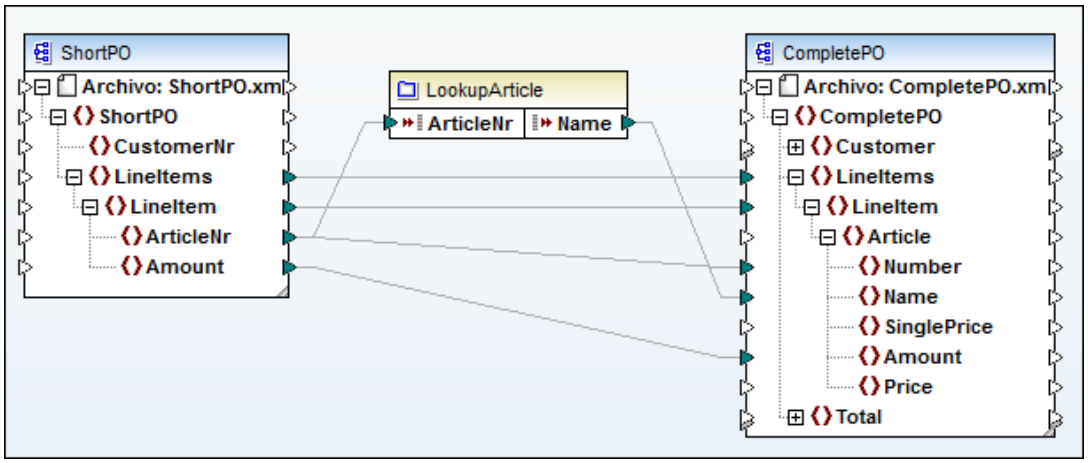


12. Arrastre la función **`LookupArticle`** hasta el panel **Asignación**.

    La función definida por el usuario aparece:
    - o con el nombre **`LookupArticle`** en la barra de título
    - o con los iconos de entrada y salida **`ArticleNr`** y **`Name`** respectivamente.



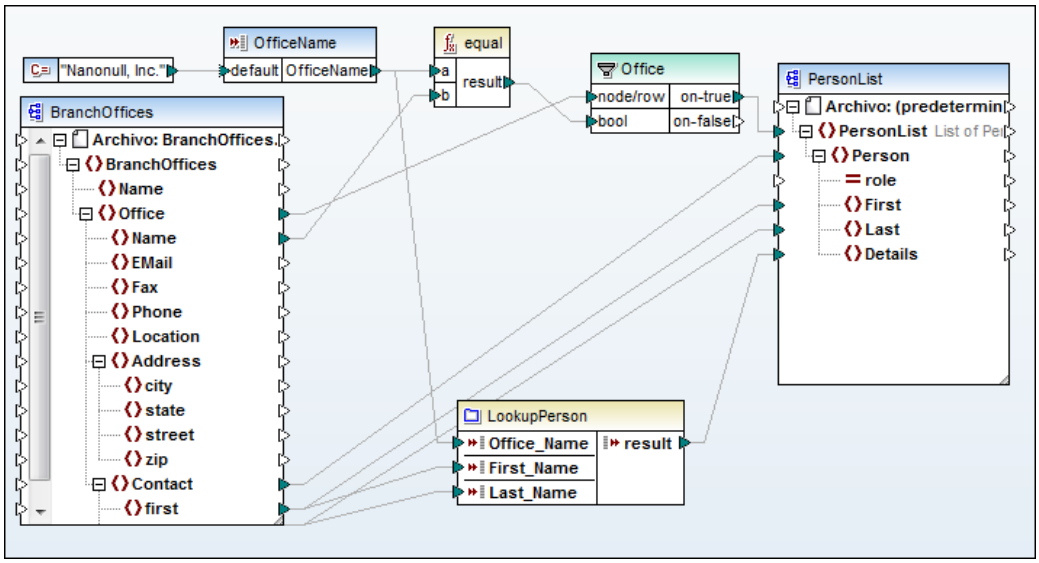
13. Cree las conexiones que aparecen en la imagen siguiente y abra el panel *Resultados* para ver el resultado de la asignación.



7.2.4 Ejemplo de función definida por el usuario

El archivo **PersonListByBranchOffice.mfd** de la carpeta **<Documentos>\Altova \MapForce2018\MapForceExamples** incluye estas características:

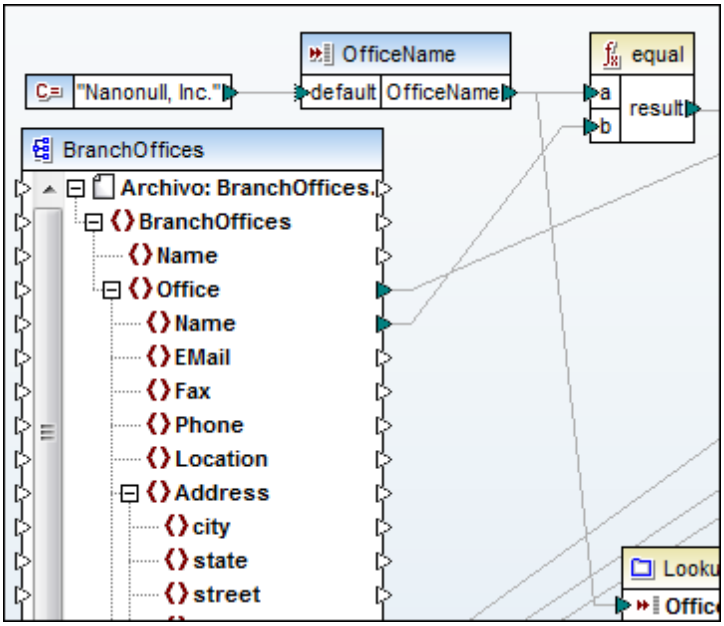
- Funciones definidas por el usuario anidadas (p. ej. **LookupPerson**).
- Funciones de búsqueda que generan un resultado de cadena (p. ej. **LookupPerson**).
- Parámetros de entrada **opcionales** que también pueden dar un valor **predeterminado** (p. ej. el componente **EqualAnd** que está en el componente LookupPerson).
- Parámetros de entrada **configurables** que también pueden hacer de parámetros de la línea de comandos cuando se ejecute el código de asignación generado.



Parámetros de entrada configurables

El componente de entrada (OfficeName) recibe datos cuando se ejecuta la asignación. Esto es posible de dos maneras:

- como parámetro de la **línea de comandos** cuando se ejecuta el código generado (p. ej. Mapping.exe /OfficeName "Nanonull Partners, Inc.")
- como valor de **vista previa** cuando se usa el motor de ejecución integrado para obtener una vista previa de los resultados en el panel *Resultados*.

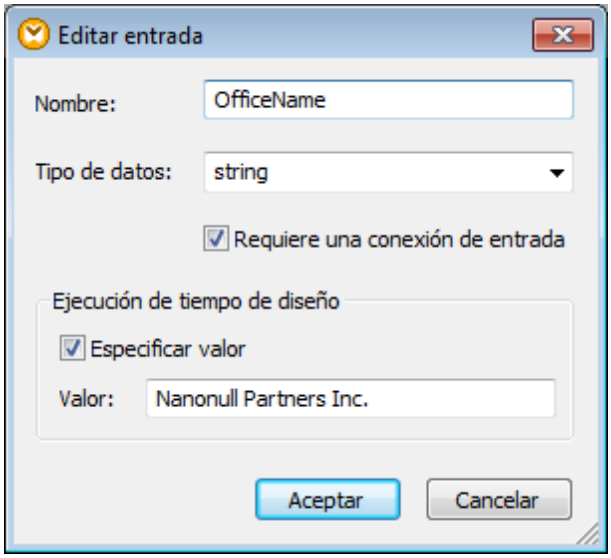


Para definir el valor de entrada:

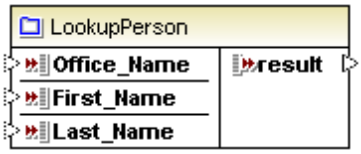
1. Haga doble clic en el componente de entrada e introduzca un valor distinto en el cuadro de texto *Valor* del grupo *Ejecución de tiempo de diseño* (p. ej. "Nanonull Partners, Inc.". Haga clic en **Aceptar** para confirmar.
2. Haga clic en el panel *Resultados* para ver el efecto.
   Ahora aparece un conjunto de personas distinto.

   Recuerde que los datos que introduzca en este cuadro de diálogo solamente se usan en el modo de vista previa (es decir, cuando se hace clic en el panel *Resultados*). Si no introduce ningún valor o si desactiva la casilla *Especificar valor*, entonces se usarán los datos que estén asignados al icono de entrada predeterminado.

   Consulte el apartado Componentes de entrada para obtener más información.



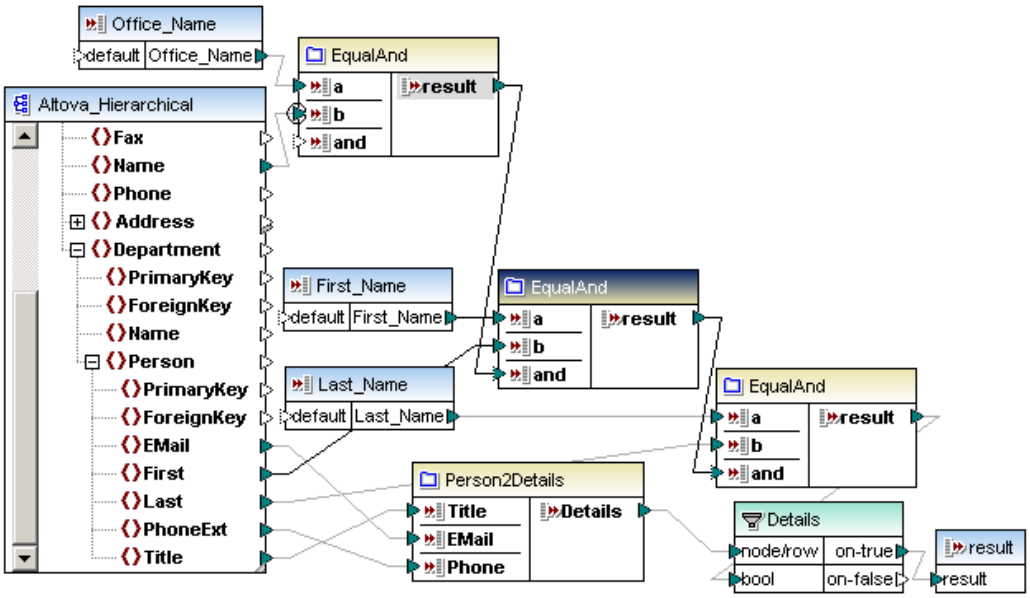
Componente LookupPerson



Si hace doble clic en este componente, podrá ver los componentes que lo constituyen (*imagen siguiente*). Este componente se encarga de:

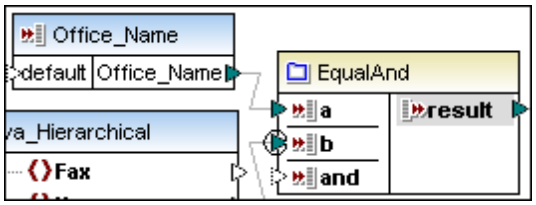
- **Comparar** los elementos Office, First y Last Name de BranchOffices.xml con los mismos campos del archivo Altova_Hierarchical.xml file (para la comparación usa los componentes de **entrada** y la función definida por el usuario **EqualAnd**).
- **Combinar** los nodos Email, PhoneExt y Title con ayuda de la función definida por el usuario **Person2Details**.
- **Pasar** los datos combinados al componente de **salida** si todas las comparaciones EqualAnd previas dan como resultado true (es decir, si pasan el valor true al componente de filtrado).

Las funciones definidas por el usuario siempre tienen un valor de salida, que incluso puede ser una cadena vacía. Este sería el caso si el valor binario del componente de filtrado fuera false. En ese caso el resultado de la función sería una cadena vacía, en lugar de los datos que suministra el componente Person2Details.



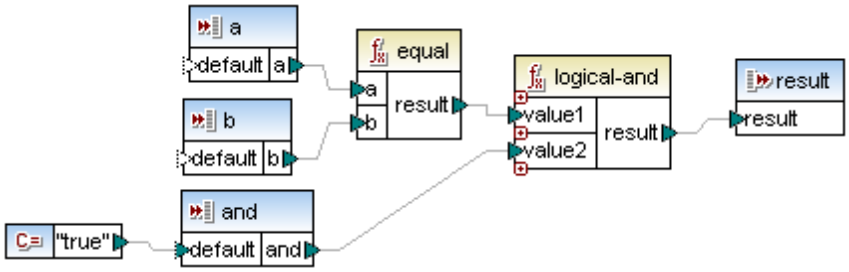
- Los tres componentes de **entrada**, Office_Name, First_Name, Last_Name, reciben datos del archivo BranchOffices.xml.
- El componente **EqualAnd** compara dos valores y suministra un valor de comparación **opcional**, así como un valor predeterminado.
- Person2Details combina tres campos personales y pasa el resultado al componente de filtrado.

Componente EqualAnd

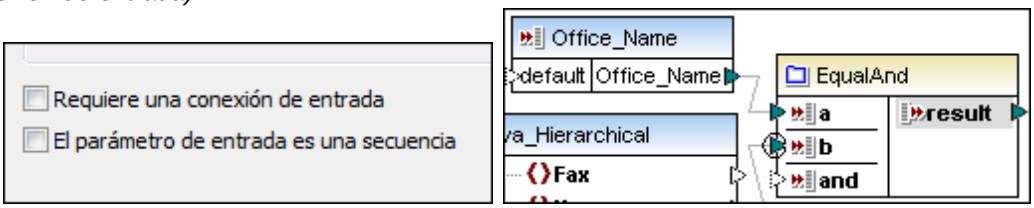


Si hace doble clic en este componente, podrá ver los componentes que lo constituyen (*imagen siguiente*). Este componente se encarga de:

- Comparar dos parámetros de entrada (**a** y **b**) y pasar el resultado al componente logical-and. Observe que el parámetro **b** se definió como **contexto prioritario**. Esto garantiza que se procesen primero los datos personales de la oficina que vienen dados por el parámetro de entrada **a** se procesen.
- Combinar en **Logical-and** el resultado de la primera comparación con un parámetro de entrada **opcional** ("and").
- Pasar el valor binario de esta comparación al parámetro de salida.

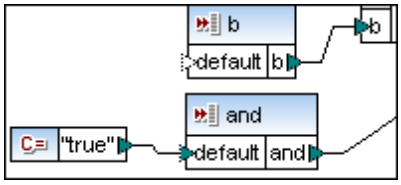


Parámetros opcionales
Si hace doble clic en el parámetro "and" de la función definida por el usuario **EqualAnd** del ejemplo anterior, podrá decidir si el parámetro es opcional o no (con la casilla *Requiere una conexión de entrada)*.



Si la casilla *Requiere una conexión de entrada* está desactivada:

- No es necesario que el icono de entrada de esta función definida por el usuario esté conectado (p. ej. el parámetro **and** de la primera función EqualAnd no tiene un conector de entrada). El icono de entrada tiene un contorno de guiones que indica que es opcional.
- Puede suministrarse un valor **predeterminado** con solo conectar un componente dentro de la función definida por el usuario (p. ej. usando un componente de constante que contenga el valor "true").



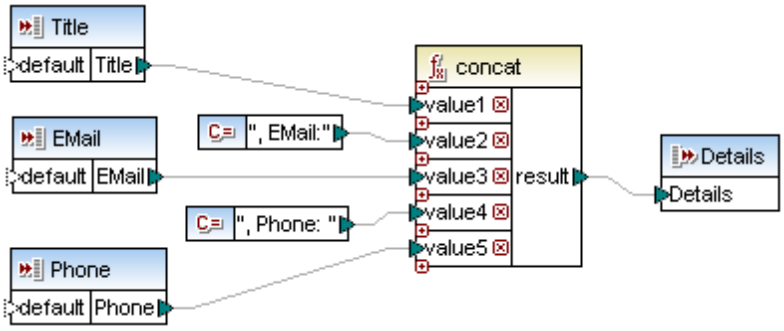
- La asignación entre otro elemento y la entrada opcional tiene prioridad sobre el valor predeterminado. Por ejemplo, el parámetro "and" de la segunda función EqualAnd recibe datos de entrada del parámetro "result" de la primera función EqualAnd definida por el usuario.

Componente Person2Details

Si hace doble clic en este componente, podrá ver los componentes que lo constituyen (*imagen*

siguiente). Este componente se encarga de:

- Encadenar tres entradas y pasar la cadena resultante al parámetro de salida.
- Puede cambiar el nombre del parámetro y seleccionar el tipo de datos con solo hacer doble clic en el parámetro de salida.



7.2.5 Función definida por el usuario compleja: nodo XML como entrada

Este apartado utiliza el archivo de ejemplo `lookup-udf-in.mfd`, disponible en la carpeta [...\MapForceExamples](). A continuación explicamos cómo se define una función definida por el usuario de tipo inline con un componente de entrada complejo.

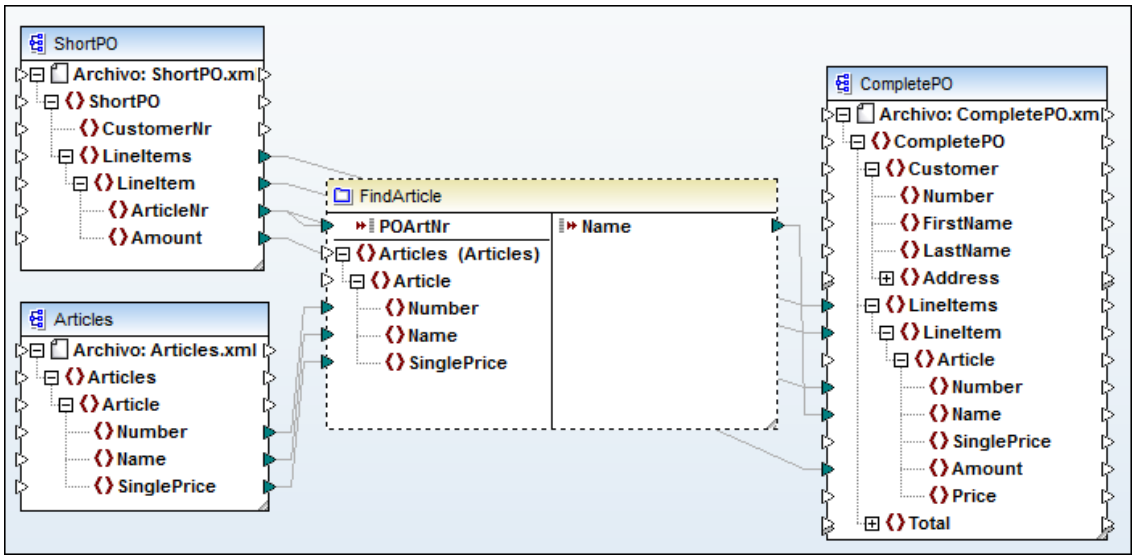
Tenga en cuenta que la función definida por el usuario **`FindArticle`** tiene dos partes.

La parte de la izquierda contiene los parámetros de entrada:

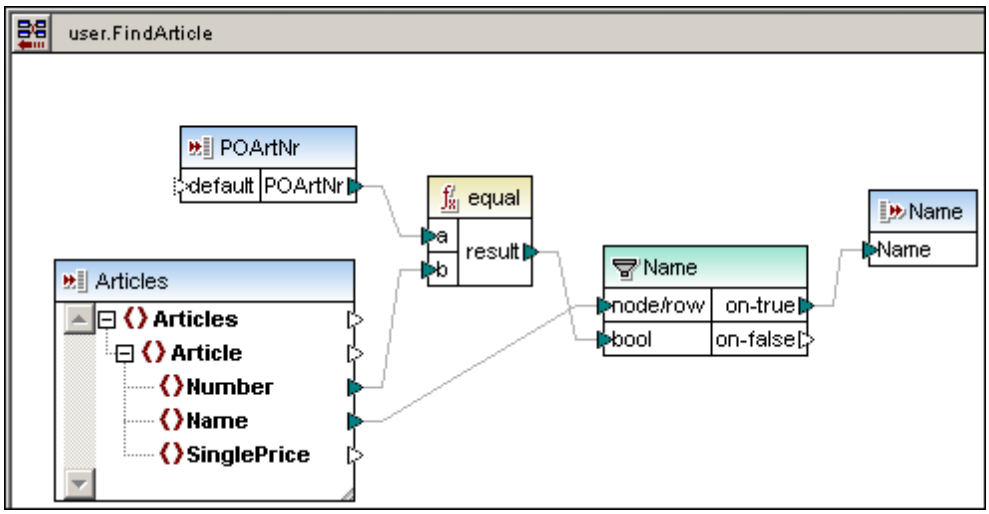
- un parámetro de entrada simple llamado **`POArtNr`**
- un parámetro de entrada complejo llamado **`Articles`**, cuyos nodos XML secundarios tienen asignaciones directas a otros componentes de la asignación

La parte de la derecha contiene:

- un parámetro de salida simple llamado **`Name`**.



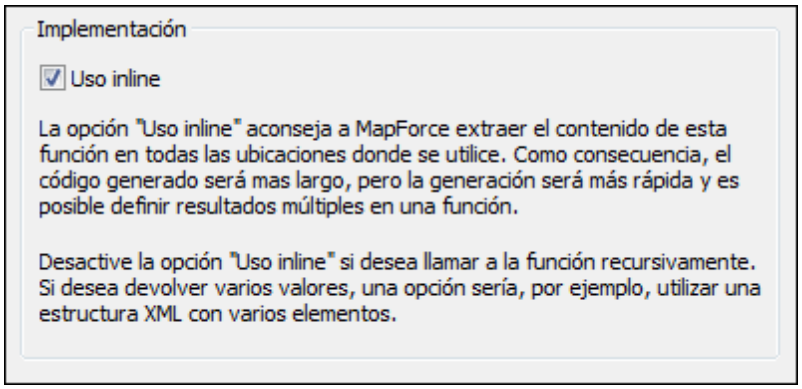
A continuación puede ver los componentes que constituyen la función definida por el usuario, con los dos componentes de entrada a la izquierda y el de salida a la derecha.



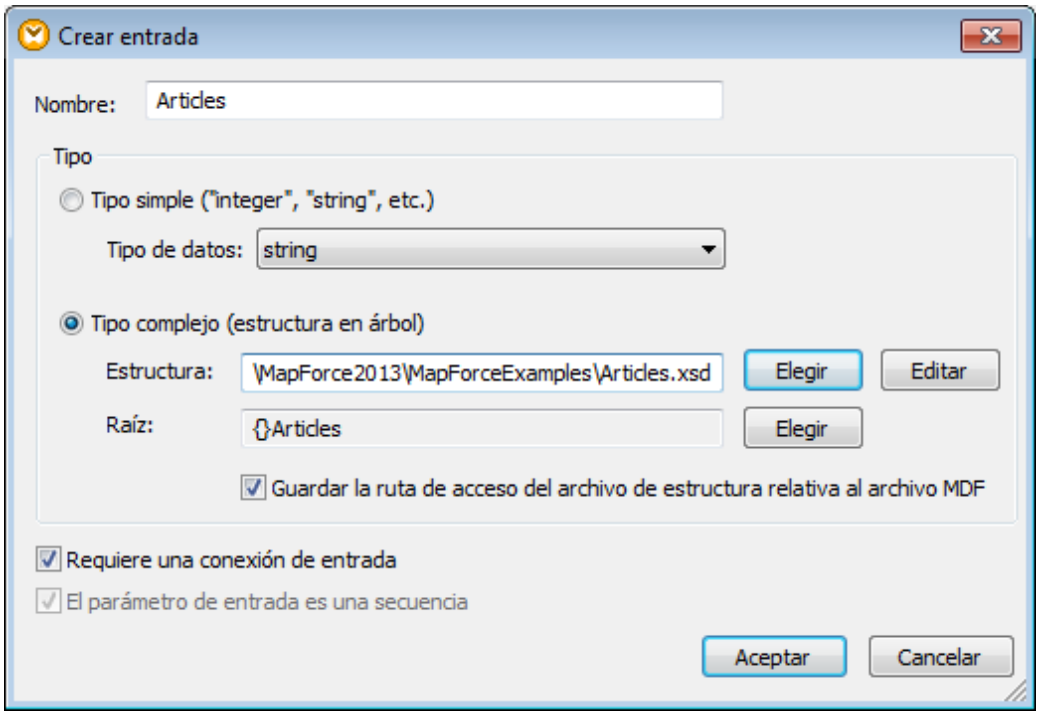
7.2.5.1 *Definir componentes de entrada complejos*

Siga estas instrucciones para crear una función que tome una estructura XML como parámetro de entrada:

1. Cree una función definida por el usuario (**Función | Crear una función definida por el usuario**). Recuerde que la opción *Uso inline* está activada por defecto.



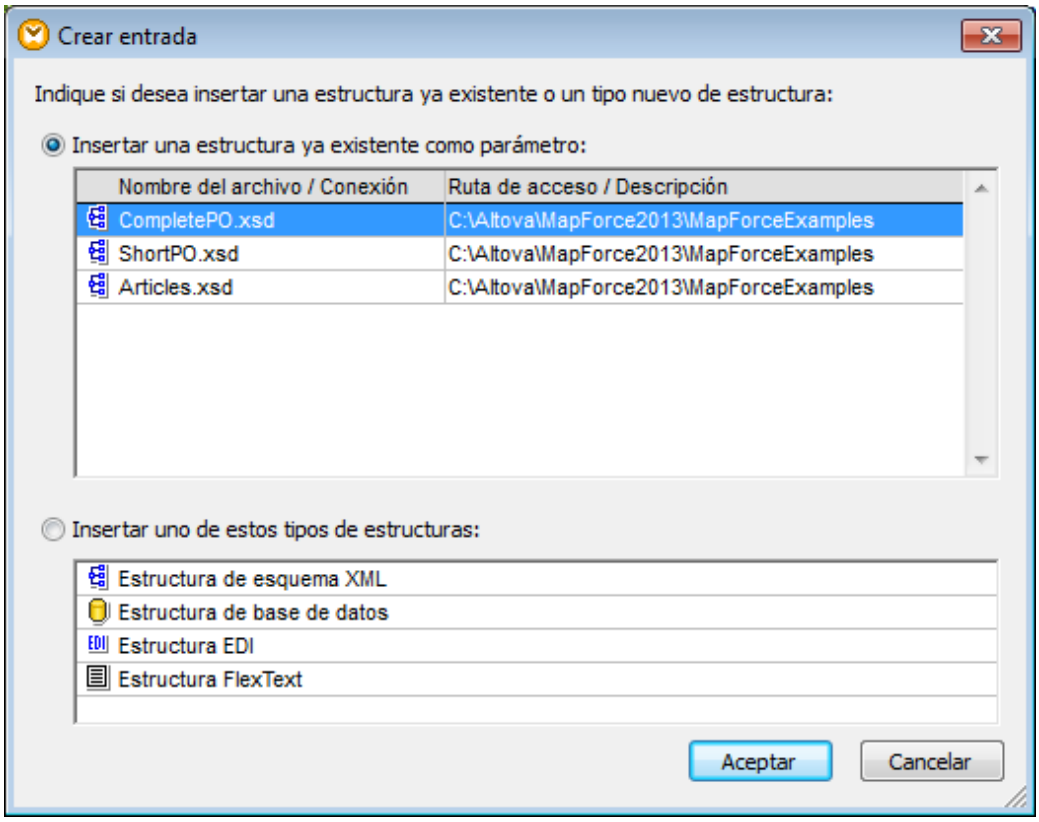
2. Haga clic en el icono **Insertar componente de entrada** de la barra de herramientas.
3. Escriba el nombre del componente de entrada en el campo *Nombre*.



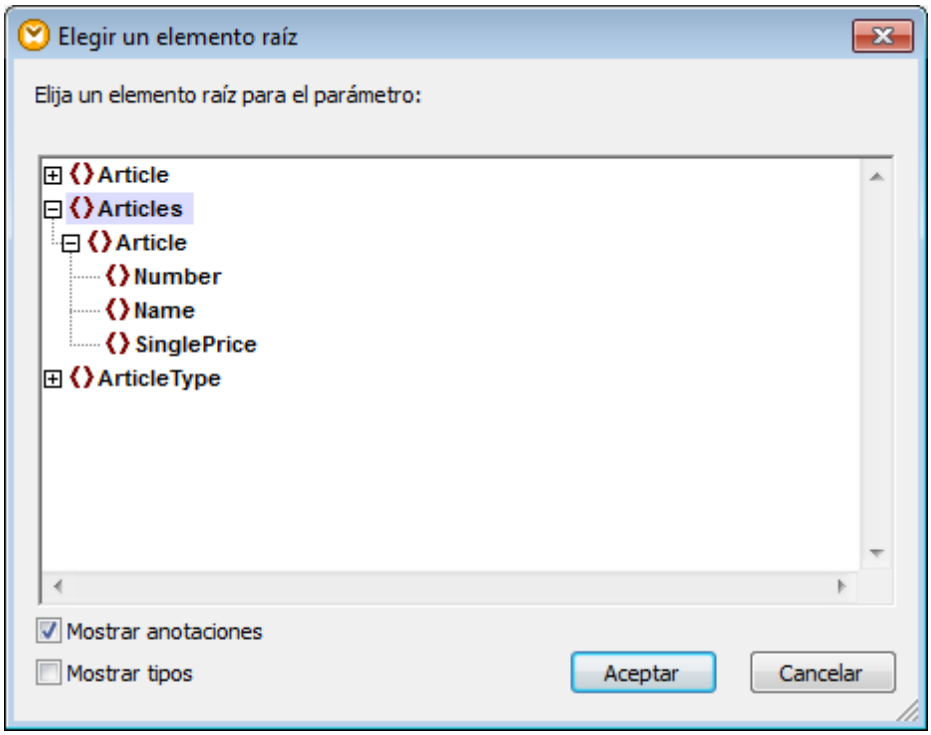
4. Seleccione el botón de opción *Tipo complejo (estructura en árbol)* y después haga clic en el botón **Elegir** del campo *Estructura*. Se abre otro cuadro de diálogo con dos cuadros de lista.

   El primer cuadro de lista enumera los componentes ya **existentes** en la asignación (en este caso, tres esquemas). Es decir, la lista enumera todos los componentes que ya se insertaron en la asignación activa (p. ej. un esquema XML).

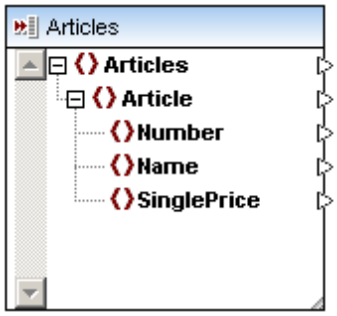
   En el cuadro de lista inferior puede seleccionar una estructura de datos compleja **nueva** (es decir, un esquema XML).



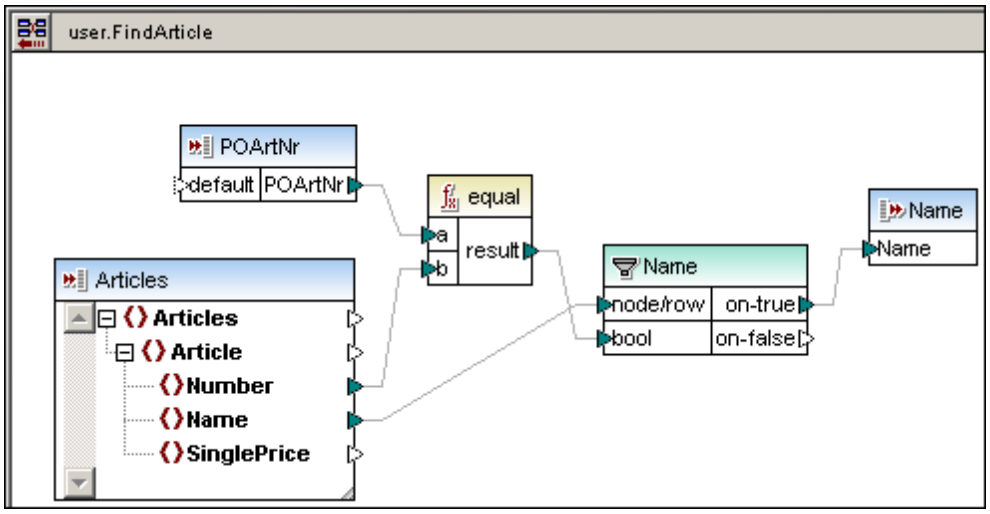
5. Seleccione el botón de opción *Insertar una estructura nueva*, después la opción **Estructura de esquema XML** y haga clic en **Aceptar**.
6. Seleccione **`Articles.xsd`** en el cuadro de diálogo Abrir.
7. Seleccione el elemento que desea usar como elemento raíz del componente (p. ej. **`Articles`**) y haga clic en **Aceptar**. Haga otra vez clic en **Aceptar** para cerrar el otro cuadro de diálogo.



El componente **Articles** se inserta en la función definida por el usuario. Observe que a la izquierda del nombre del componente aparece el icono de entrada . Esto indica que el componente se usa como componente de entrada complejo.

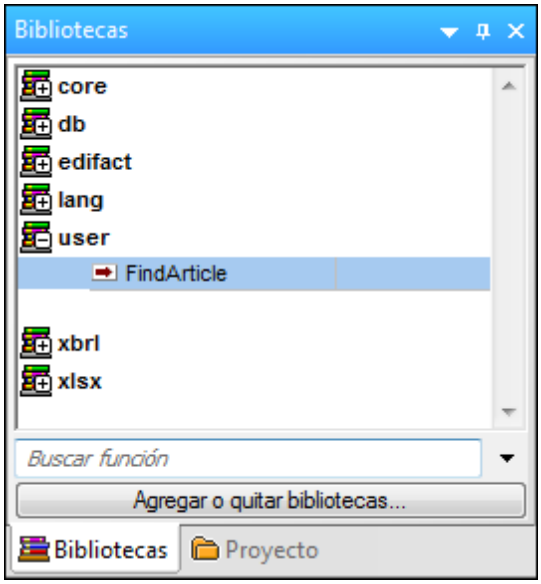


8. Inserte el resto de componentes que aparecen en la imagen siguiente, es decir, otro componente de entrada simple (**POArtNr**), un filtro, la función **equal** y un componente de salida (**Name**). Después conéctelos tal y como muestra la imagen.

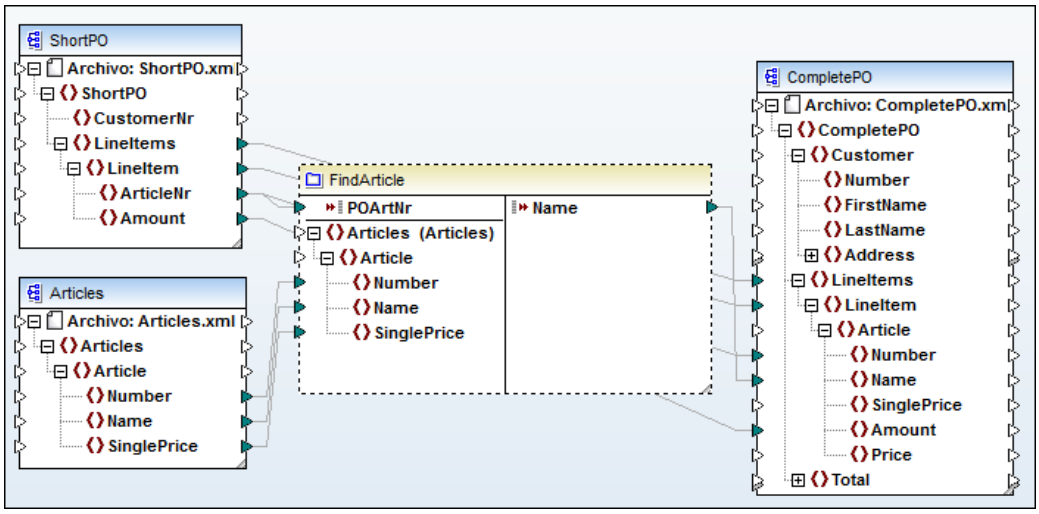


- el componente de entrada **`Articles`** recibe sus datos de **fuera** de la función. Los iconos de entrada que permiten la asignación de datos a este componente están ahí.
- a los componentes de entrada complejos no se les puede asignar un archivo XML de instancia para suministrar datos desde la función definida por el usuario.
- el componente de entrada **`POArtNr`** aporta los datos de número de artículo de **`ShortPO`** con los que se deben comparar los datos de **`Article | Number`**.
- el filtro filtra los registros cuyos números son idénticos y los envía al componente de salida.

9. Haga clic en el icono **Volver** para volver a la asignación principal.
10. Inserte la función definida por el usuario nueva (arrastrándola desde la ventana Bibliotecas hasta el panel *Asignación*).



11. Cree las conexiones que aparecen en esta imagen.



La parte de la izquierda contiene los parámetros de entrada a los que se asignan elementos de los dos archivos XML:

- **ShortPO** aporta los datos para el parámetro de entrada **`POArtNr`**.
- **Articles** aporta los datos para el parámetro de entrada complejo. El archivo de instancia `Articles.xml` se asignó al archivo de esquema **Articles** cuando se insertó el componente.
- El parámetro de entrada complejo **`Articles`** con sus nodos XML secundarios, a los que se asignaron datos desde el componente **Articles**.

La parte de la derecha contiene:

- un parámetro de entrada simple llamado **`Name`**, que pasa los elementos filtrados que tienen el mismo número de **`Article`** al elemento **`Name`** del componente de destino **CompletePO**.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <CompletePO xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3    <LineItems>
4      <LineItem>
5        <Article>
6          <Number>3</Number>
7          <Name>Pants</Name>
8          <Amount>5</Amount>
9        </Article>
10     </LineItem>
11     <LineItem>
12       <Article>
13         <Number>1</Number>
14         <Name>T-Shirt</Name>
15         <Amount>17</Amount>
16       </Article>
17     </LineItem>
18   </LineItems>
19 </CompletePO>
```

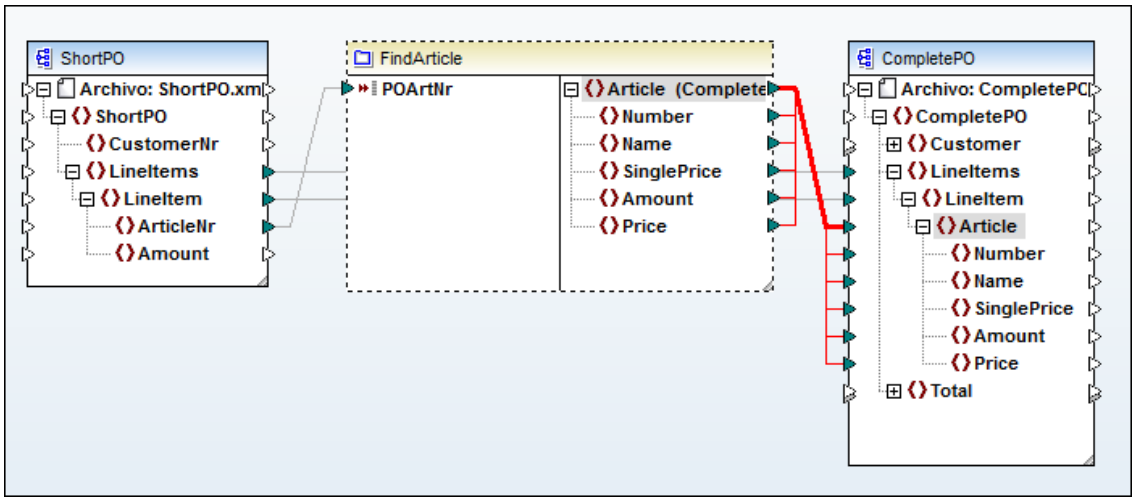
Nota: cuando cree conexiones de **copia total** entre un esquema y un parámetro de una función definida por el usuario, ambos componentes deben estar basados en el mismo esquema. Sin embargo, no es necesario que ambos tengan el mismo elemento raíz.

7.2.6 Función definida por el usuario compleja: nodo XML como salida

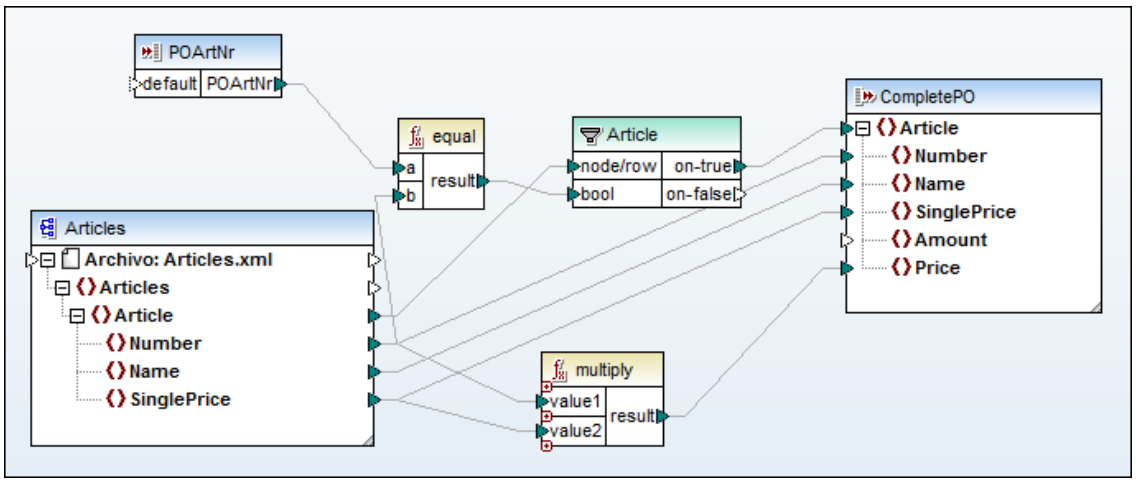
Este apartado utiliza el archivo de ejemplo `lookup-udf-out.mfd`, disponible en la carpeta **...\MapForceExamples** A continuación explicamos cómo se define una función definida por el usuario de tipo inline con un componente de salida complejo.

Tenga en cuenta que la función definida por el usuario FindArticle tiene dos partes:

- La parte de la izquierda contiene un parámetro de entrada simple llamado **`POArtNr`**
- La parte de la derecha contiene un parámetro de salida complejo llamado **`Article (CompletePO)`**, cuyos nodos XML secundarios están asignados al componente de destino **CompletePO**.



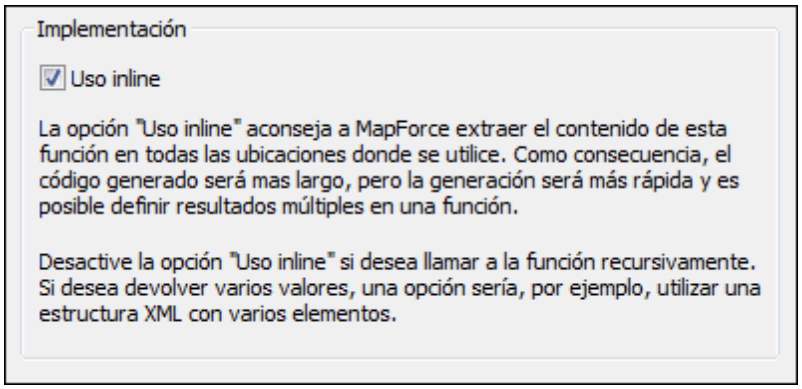
A continuación puede ver los componentes que constituyen la función definida por el usuario, con los dos componentes de entrada a la izquierda y el de salida a la derecha.



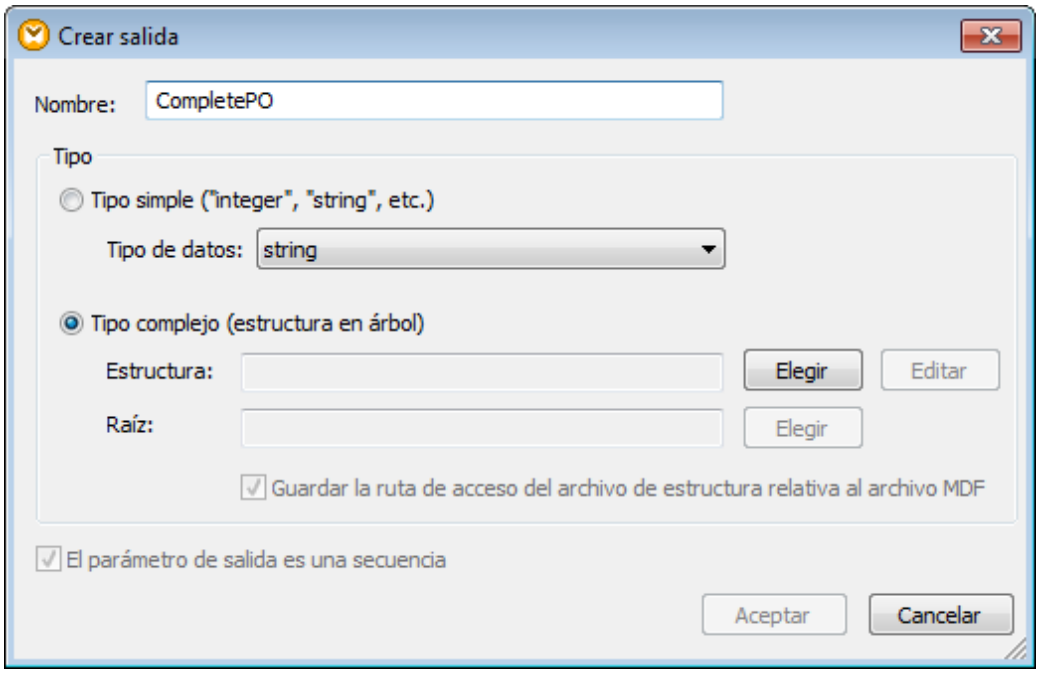
7.2.6.1 Definir componentes de salida complejos

Siga estas instrucciones para crear una función que devuelva una estructura XML como parámetro de salida:

1. Cree una función definida por el usuario (**Función | Crear una función definida por el usuario**). Recuerde que la opción *Uso inline* está activada por defecto.



2. Haga clic en el icono **Insertar componente de salida** de la barra de herramientas. En el cuadro de diálogo escriba el nombre del componente de salida en el campo *Nombre* (p. ej. CompletePO).

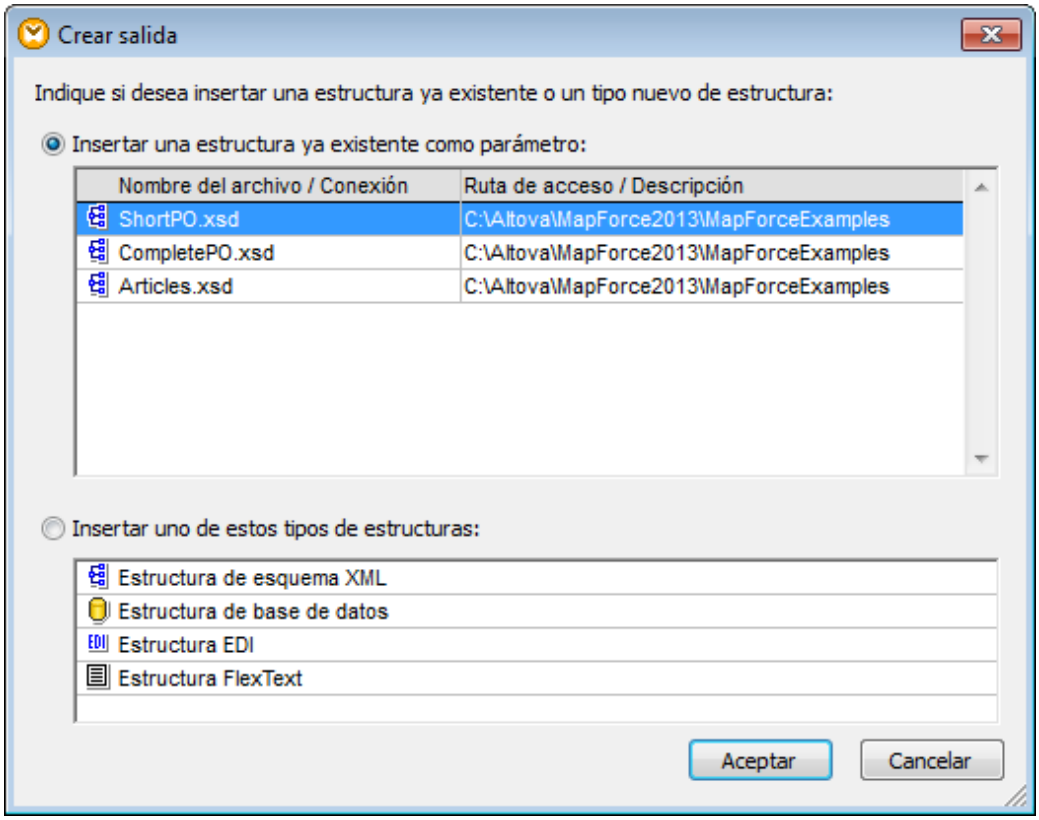


3. Seleccione el botón de opción *Tipo complejo (estructura en árbol)* y después haga clic en el botón **Elegir** del campo *Estructura*. Se abre otro cuadro de diálogo con dos cuadros de listas.

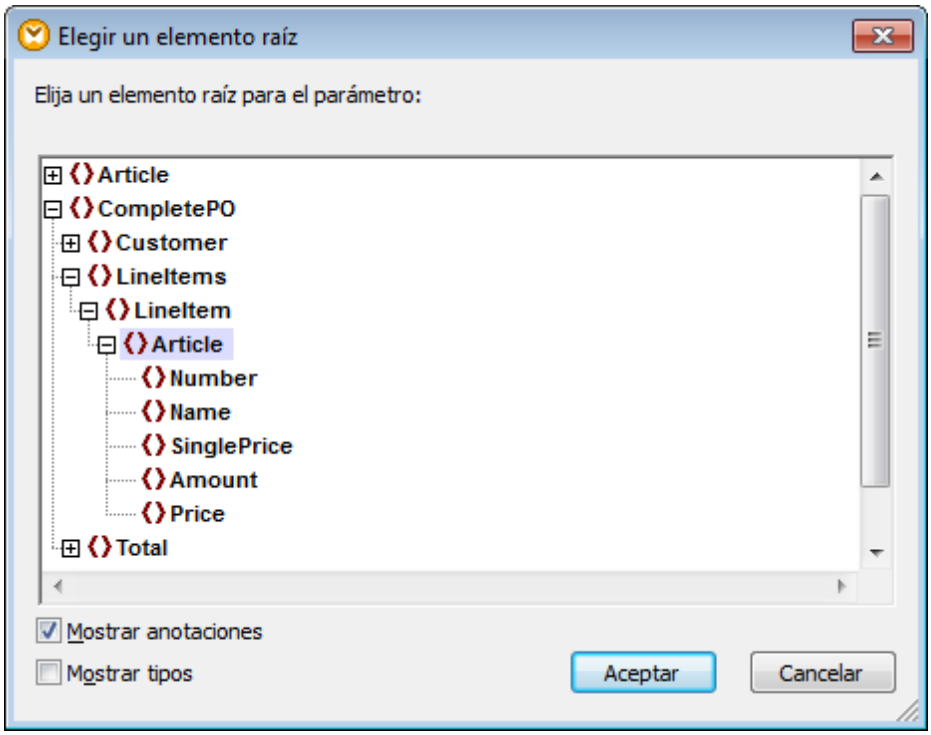
   El primer cuadro de lista enumera los componentes ya **existentes** en la asignación (en

este caso, tres esquemas). Es decir, la lista enumera todos los componentes que ya se insertaron en la asignación activa (p. ej. un esquema XML).

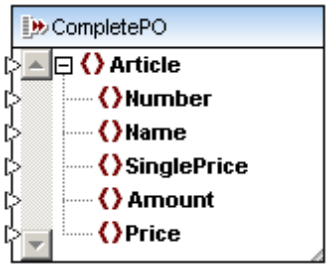
En el cuadro de lista inferior puede seleccionar una estructura de datos compleja **nueva** (es decir, un esquema XML).



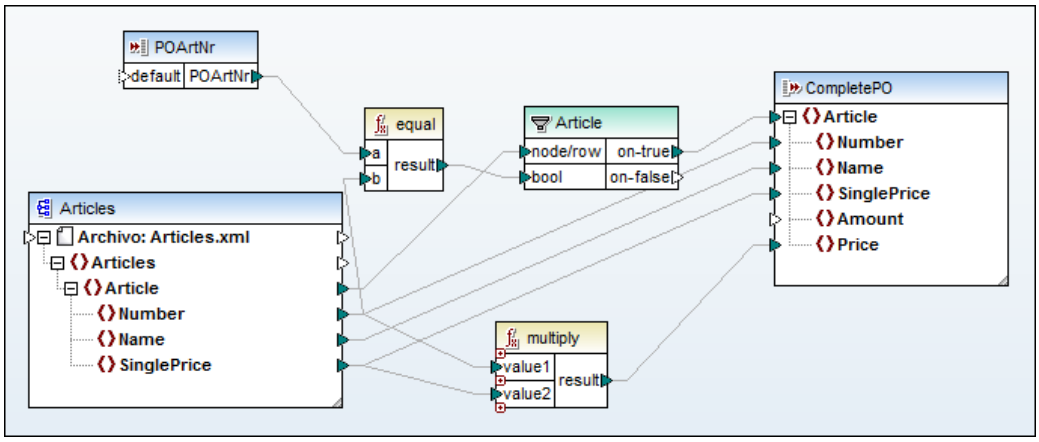
4. Seleccione el botón de opción *Insertar una estructura nueva*, seleccione la opción **Estructura de esquema XML** y haga clic en **Aceptar**.
5. Seleccione el esquema `CompletePO.xsd` en el cuadro de diálogo Abrir.
6. Haga clic en el elemento que desea usar como elemento raíz del componente (p. ej. **`Article`**) y haga clic en **Aceptar**. Haga otra vez clic en **Aceptar** para cerrar el otro cuadro de diálogo.



El componente **CompletePO** se inserta en la función definida por el usuario. Observe que a la izquierda del nombre del componente aparece el icono de salida . Esto indica que el componente se utiliza como parámetro/componente de salida complejo.

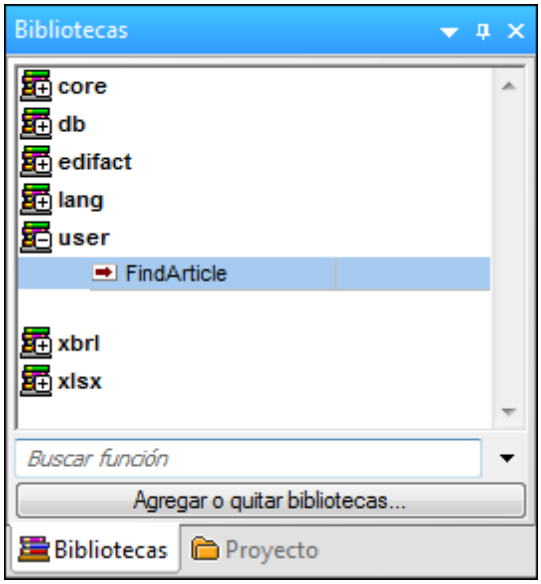


7. Inserte el esquema XML **Articles** en la función definida por el usuario y asigne el archivo `Articles.xml` como XML de instancia.
8. Inserte el resto de componentes que aparecen en la imagen siguiente, es decir: el componente de entrada simple **`POArtNr`**, un filtro, una función **`equal`** y una función **`multiply`**. Después conéctelas como muestra la imagen.

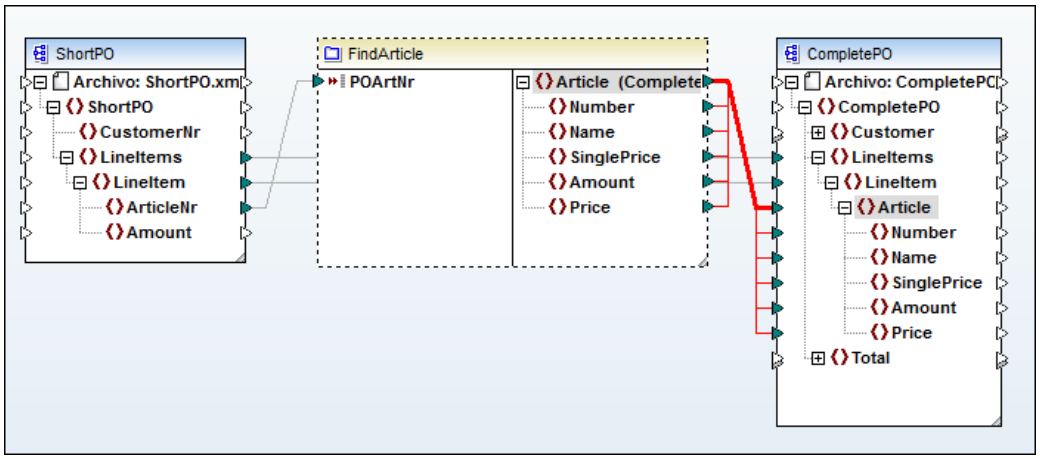


- el componente **Articles** recibe datos del archivo de instancia `Articles.xml`, situado dentro de la función definida por el usuario.
- los componentes de entrada aportan los datos de **`POArtNr`** y **`Amount`**, con los que se comparan los datos de **`Articles | Number & Price`**.
- el filtro filtra los registros que tienen números idénticos y los pasa al componente de salida **CompletePO**.

9. Haga clic en el icono **Volver**  para volver a la asignación principal.
10. Inserte la función definida por el usuario nueva (arrastrándola desde la ventana Bibliotecas hasta el panel *Asignación*).



11. Cree las conexiones que aparecen en esta imagen. Tras crear el conector entre el parámetro de salida **`Article (CompletePO)`** y el componente de destino, haga clic con el botón derecho en el conector y seleccione **Copia total** en el menú contextual. Los demás conectores se generan automáticamente y se resaltan en rojo (*imagen siguiente*).



Nota: cuando cree conexiones de **copia total** entre un esquema y un parámetro de una función definida por el usuario, ambos componentes deben estar basados en el mismo esquema. Sin embargo, no es necesario que ambos tengan el mismo elemento raíz.

La parte izquierda de la función contiene el parámetro de entrada, al que se asigna un solo elemento:

- **ShortPO** aporta el número de artículo al parámetro de entrada **`POArtNr`**.

La parte derecha de la función contiene:

- un parámetro de salida complejo llamado **`Article (CompletePO)`** con nodos XML secundarios, que asigna los elementos filtrados (los que tienen el mismo número de artículo) al componente de destino **CompletePO**.

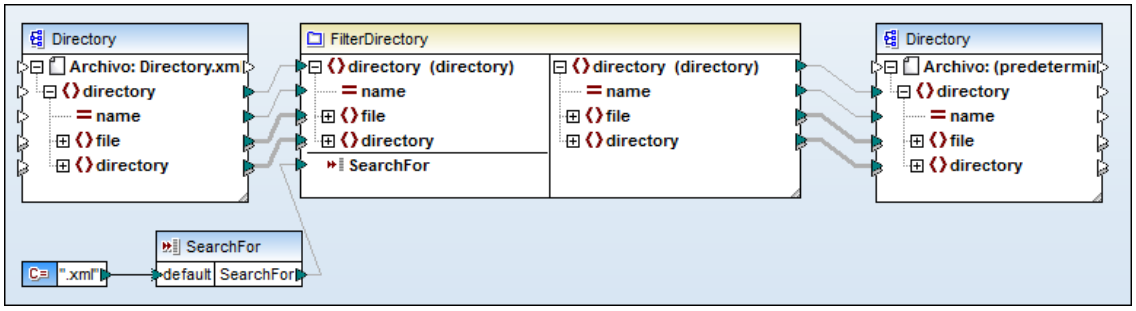
```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <CompletePO xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespace
3    <LineItems>
4      <LineItem>
5        <Article>
6          <Number>3</Number>
7          <Name>Pants</Name>
8          <SinglePrice>34</SinglePrice>
9          <Price>102</Price>
10       </Article>
11     </LineItem>
12     <LineItem>
13       <Article>
14         <Number>1</Number>
15         <Name>T-Shirt</Name>
16         <SinglePrice>25</SinglePrice>
17         <Price>25</Price>
18       </Article>
```

7.2.7 Asignación recursiva definida por el usuario

En este apartado explicamos cómo se creó la asignación **RecursiveDirectoryFilter.mfd** de la carpeta ...**\MapForceExamples** y cómo diseñar asignaciones de datos recursivas. La carpeta de proyecto MapForceExamples contiene algunos ejemplos más de asignaciones de datos

recursivas.

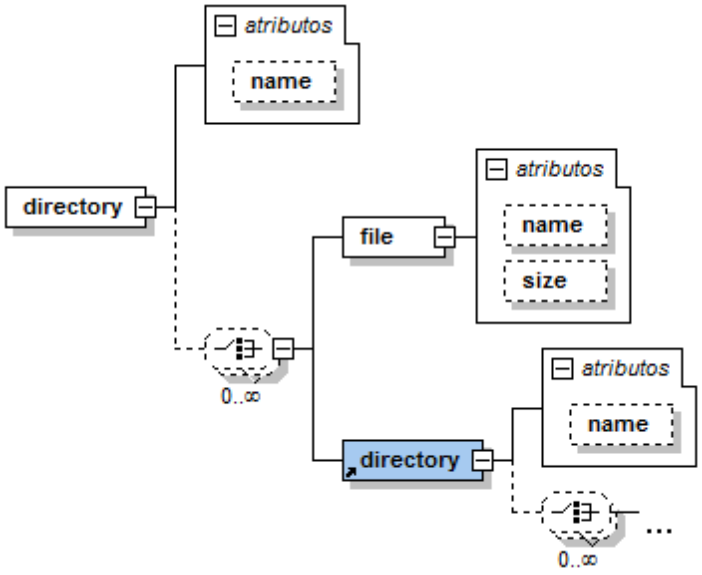
En la imagen siguiente puede ver la asignación final que contiene la función recursiva definida por el usuario **FilterDirectory**. El objetivo de esta función es filtrar una lista de archivos .xml del archivo de origen.



El **archivo de origen** que contiene los datos sobre los archivos y directorios para esta asignación es Directory.xml. Este archivo XML incluye los datos sobre directorios y archivos de forma jerárquica:

```
24      <directory name="output">
25        <file name="examplesite1.css" size="3174"/>
26        <directory name="images">
27          <file name="blank.gif" size="88"/>
28          <file name="block_file.gif" size="13179"/>
29          <file name="block_schema.gif" size="9211"/>
30          <file name="nav_file.gif" size="60868"/>
31          <file name="nav_schema.gif" size="6002"/>
32        </directory>
33      </directory>
34    </directory>
35    <directory name="Import">
36      <file name="altova.mdb" size="266240"/>
37      <file name="Data_shape.mdb" size="225280"/>
38    </directory>
39    <directory name="IndustryStandards">
40      <directory name="News">
41        <file name="high-tide.jpg" size="10793"/>
42        <file name="Newsml-example.xml" size="5004"/>
43        <file name="nitf-example.xml" size="9327"/>
44      </directory>
```

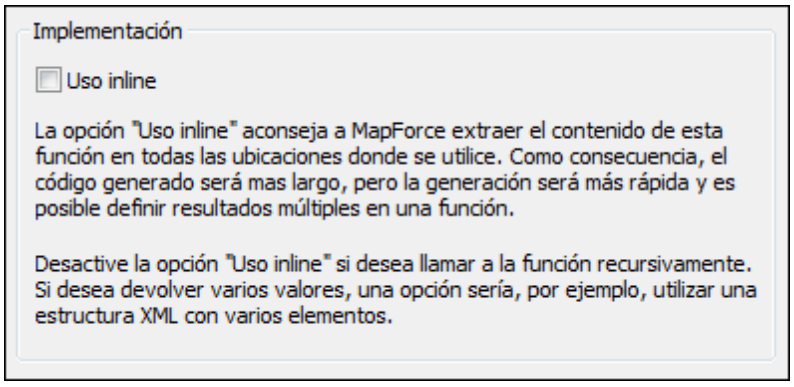
El archivo de esquema XML al que hace referencia el archivo Directory.xml tiene un elemento recursivo llamado "directory" que permite un número indeterminado de subdirectorios y de archivos debajo del elemento directory.



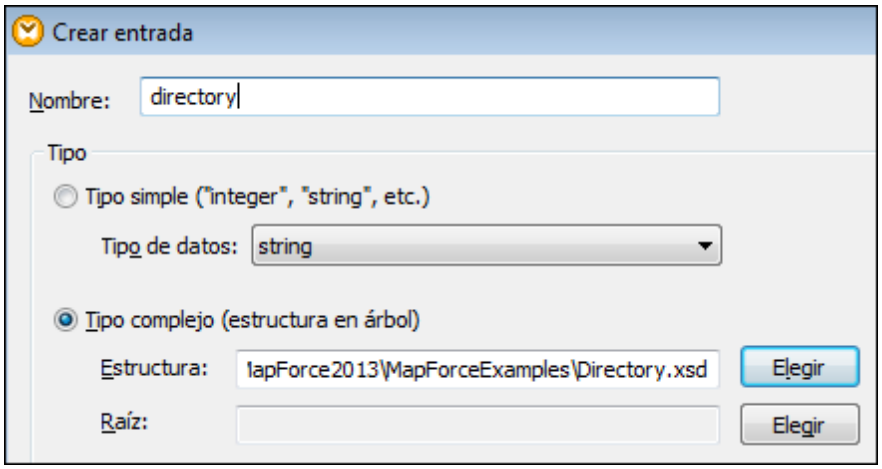
7.2.7.1 Crear una función recursiva definida por el usuario

Siga estas instrucciones para crear una función recursiva definida por el usuario:

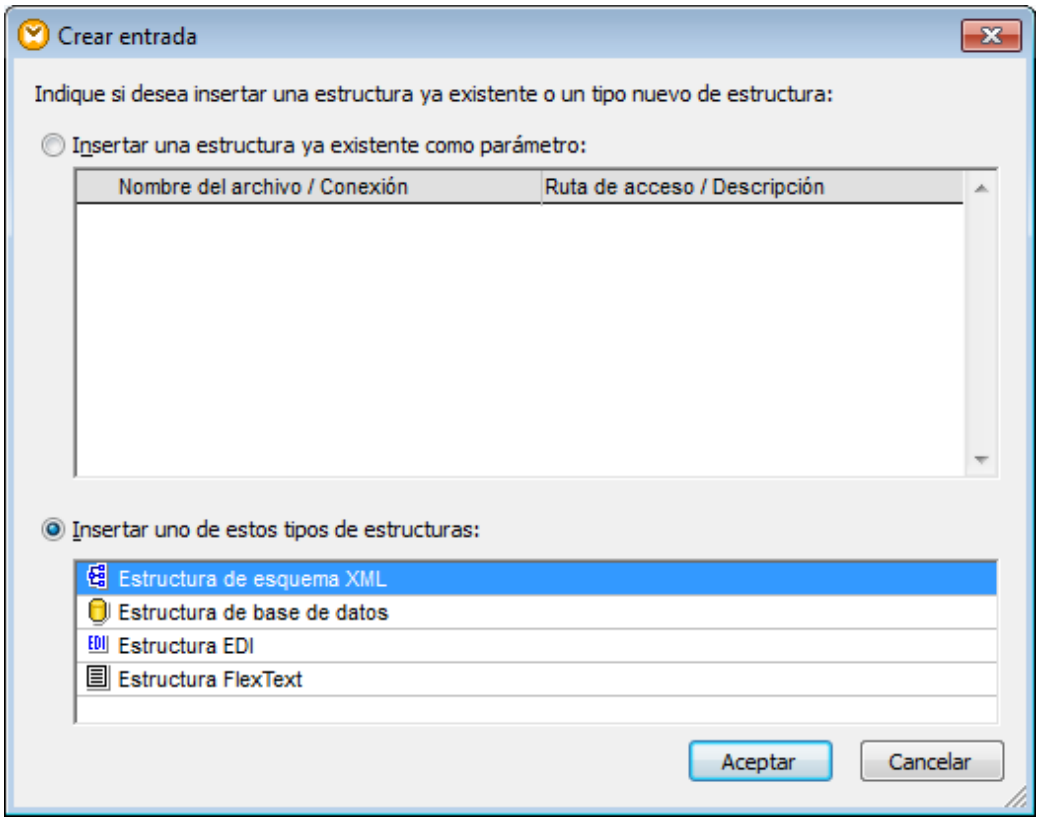
1. Cree una función definida por el usuario (**Función | Crear una función definida por el usuario**) llamada FilterDirectory.
2. Asegúrese de desactivar la casilla *Uso inline* del grupo de opciones *Implementación* para que la función sea recursiva. Haga clic en **Aceptar**.



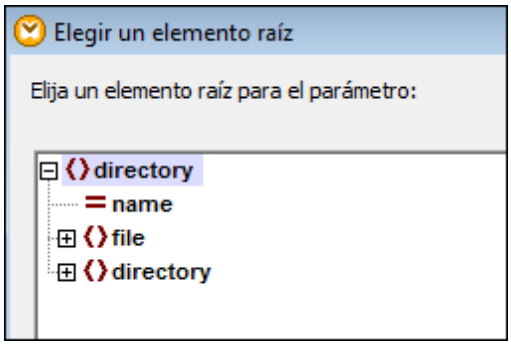
   Ahora estamos en la ventana **FilterDirectory, donde podemos crear la función definida por el usuario.**
3. Seleccione **Función | Insertar componente de entrada**.
4. Defina el nombre de componente "directory" y haga clic en el botón de opción *Tipo complejo (estructura en árbol)*.



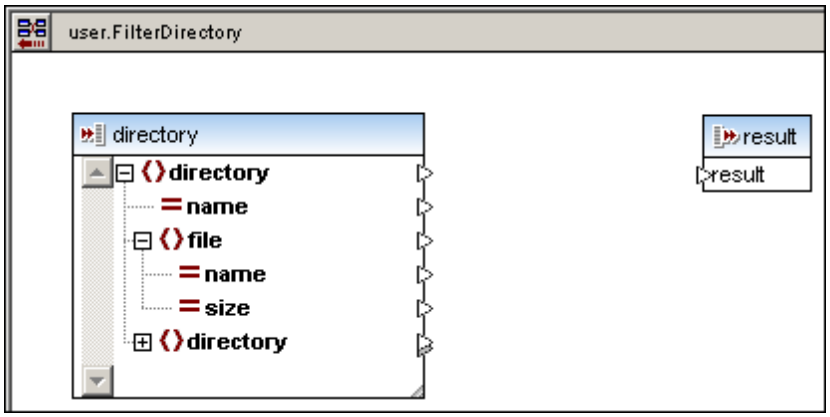
5. Haga clic en el botón **Elegir** y después en la entrada **`Estructura de esquema XML`** del panel inferior. Después haga clic en **Aceptar**.



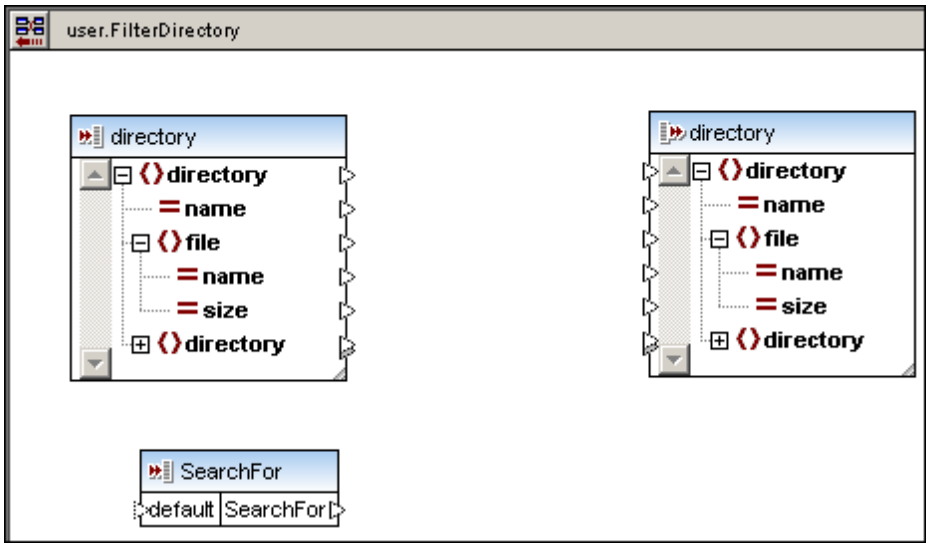
6. Seleccione el archivo **Directory.xsd** de la carpeta ...\MapForceExamples y haga clic en el botón **Abrir**.
7. Cuando la aplicación solicite el elemento raíz haga clic en **Aceptar** (el elemento raíz debería ser `directory`).



8. Haga clic en **Aceptar** para insertar el parámetro de entrada complejo. Esta es la función definida por el usuario que acabamos de crear:



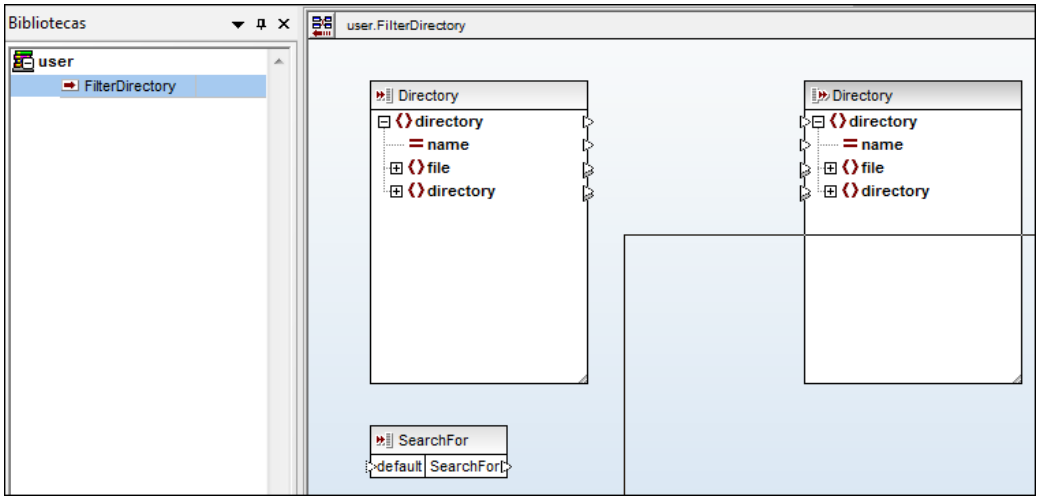
9. Elimine el componente de salida simple **result** (porque aquí debemos insertar un componente de salida complejo).
10. Seleccione **Función | Insertar componente de salida...** para insertar un componente de salida y use el método de los pasos anteriores para que el componente de salida "directory" sea de tipo complejo. Ahora tenemos dos componentes complejos: uno de entrada y otro de salida.
11. Seleccione **Función | Insertar componente de entrada...** para insertar un componente de entrada de tipo simple llamado "SearchFor". Desactive la casilla *Requiere una conexión de entrada*.



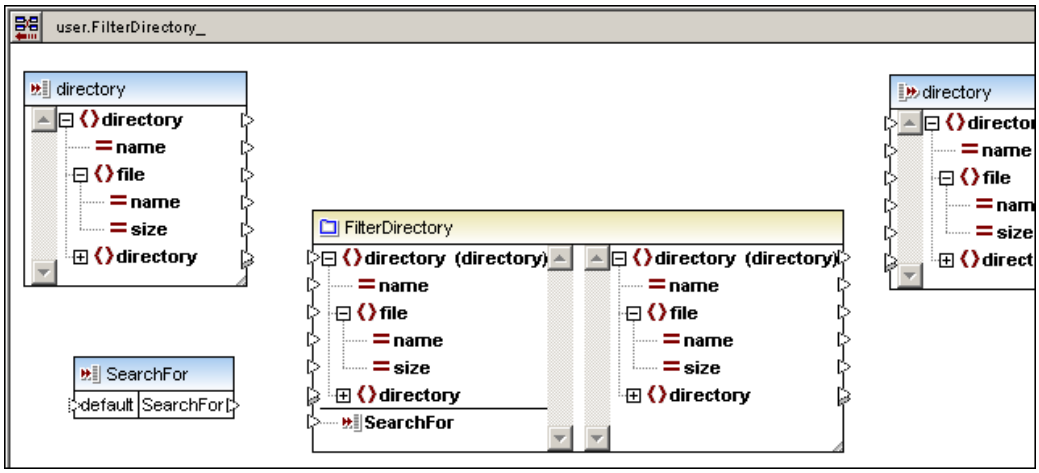
Insertar la función recursiva definida por el usuario

Llegados a este punto todos los componentes de entrada y salida necesarios están definidos para la función definida por el usuario. Ahora falta insertar la función en la ventana actual de la función definida por el usuario.

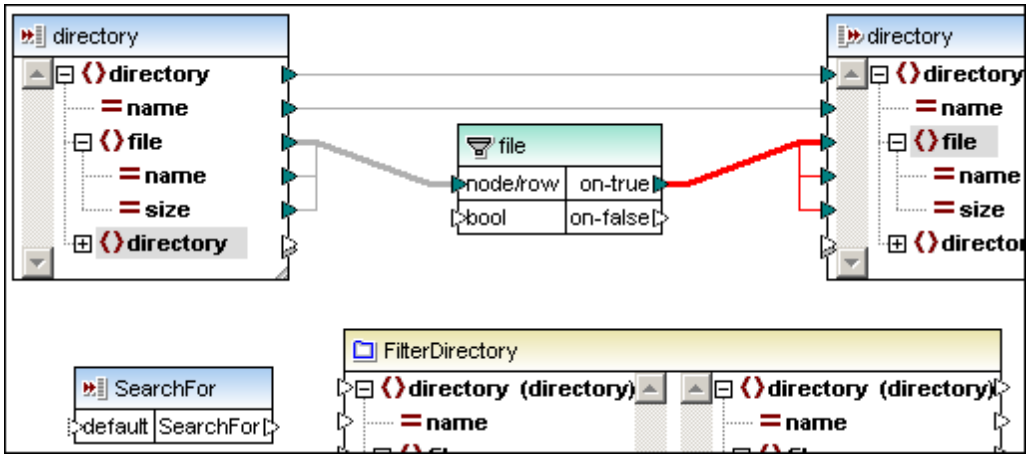
1. Busque la función "FilterDirectory" en la sección **user** de la ventana Bibliotecas.
2. Haga clic en la función FilterDirectory y arrástrela hasta la ventana de FilterDirectory en la que ha estado trabajando hasta ahora.



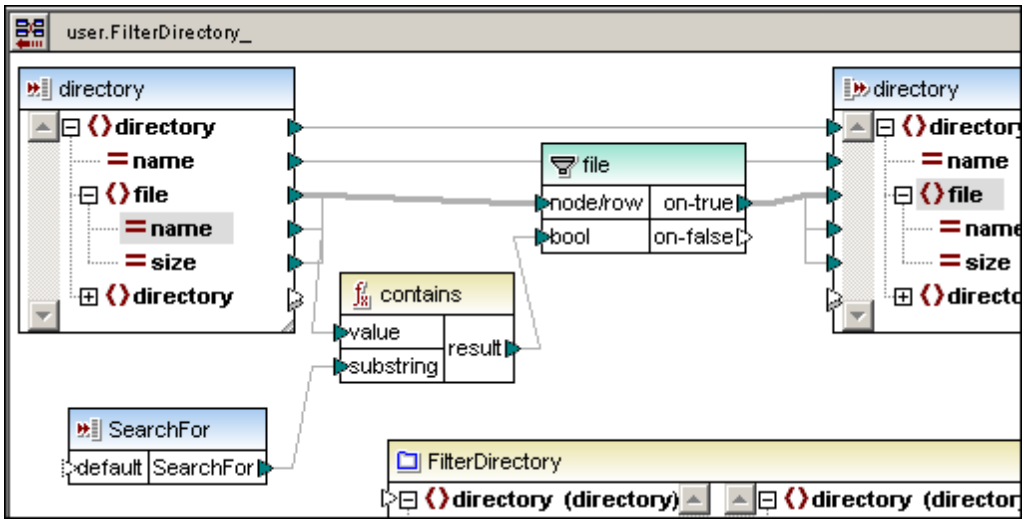
La función definida por el usuario aparece en la ventana de la función definida por el usuario en forma de componente recursivo.



3. Conecte los nodos **directory, name** y **file** del componente de entrada con los nodos del mismo nombre del componente de salida.
4. Haga clic con el botón derecho en el conector que une los nodos **file** y seleccione **Insertar filtro** en el menú contextual para introducir un componente de filtrado.
5. Haga clic con el botón derecho en el conector **on-true** y seleccione **Copia total** en el menú contextual.
   Ahora los conectores aparecen como conectores de copia total.



6. Inserte una función **contains** de la biblioteca **Core | String functions**.
7. Conecte **name** con **value** y el parámetro **SearchFor** con **substring**. Después conecte el resultado con el nodo **bool** del filtro.

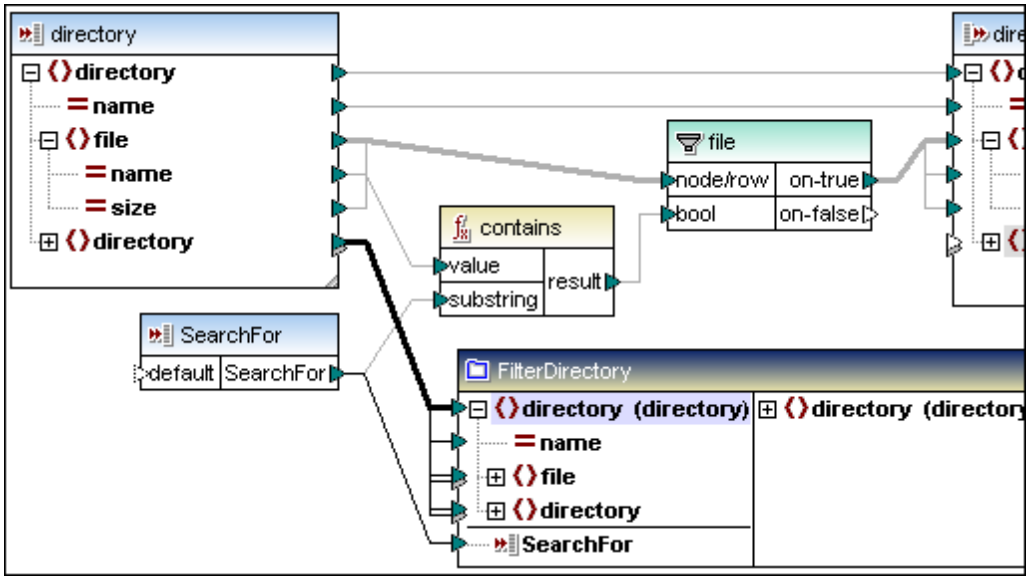


8. Para terminar conecte el nodo **SearchFor** del componente de entrada con el nodo **SearchFor** de la función definida por el usuario.

Definir la recursión

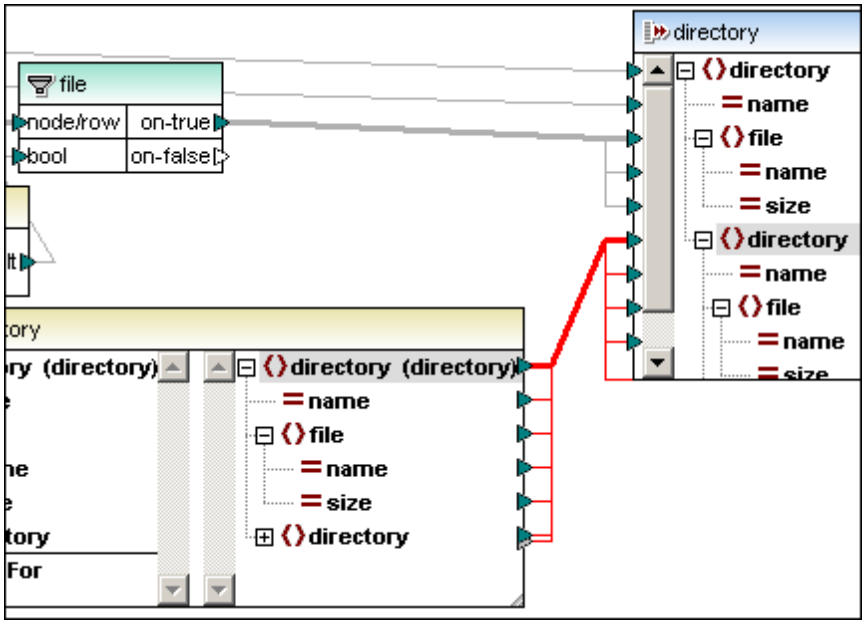
Llegados a este punto la asignación de un nivel de recursión en un solo directorio está terminada. Ahora falta definir el procesamiento del subdirectorio.

1. Para empezar asegúrese de que el icono **Alternar la conexión automática de secundarios** está activado en la barra de herramientas.
2. Conecte el nodo inferior **directory** del componente de entrada con el nodo superior **directory** de la función recursiva definida por el usuario.



3. Conecte el nodo superior de salida **directory** de la función definida por el usuario con el nodo inferior **directory** del componente de salida.

4. Haga clic con el botón derecho en el conector superior, seleccione **Copia total** en el menú contextual y haga clic en **Aceptar** si desea crear una conexión de copia total.

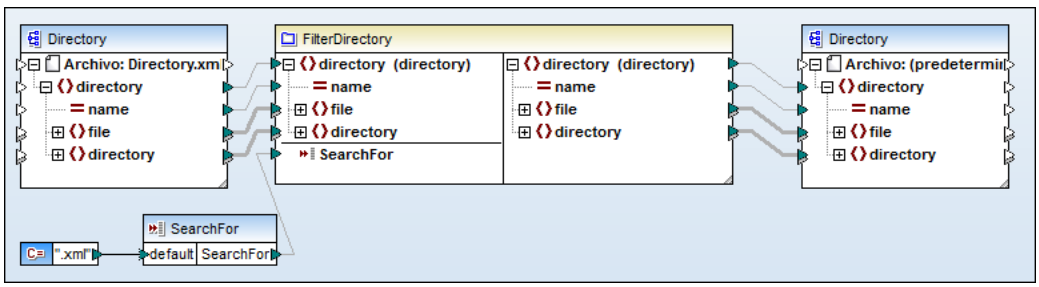


La función definida por el usuario ya está terminada.

Ahora haga clic en **Volver** para seguir diseñando la asignación en la ventana principal.

Ventana principal de asignación

1. Arrastre la función **FilterDirectory** de la sección **user** de la ventana Bibliotecas hasta el área de asignación principal.
2. Seleccione **Insertar | Archivo de esquema XML** para insertar el esquema Directory.xsd y después seleccione el archivo de instancia Directory.xml.
3. Use el mismo método para insertar un componente de salida basado en el esquema Directory.xsd y haga clic en **Omitir** cuando la aplicación solicite un archivo de instancia.
4. Inserte un componente de constante y después un componente de entrada (p. ej. SearchFor).
5. Cree las conexiones que aparecen en la imagen siguiente.
6. Cuando conecte los conectores de nivel superior, de **Directory** a **Directory**, a ambos lados del componente definido por el usuario, haga clic con el botón derecho en el conector y seleccione **Copia total** en el menú contextual.



7. Haga clic en el panel *Resultados* para ver el resultado de la asignación.

Nota: si hace doble clic en el nodo inferior directory del componente **Directory**, se abre un nuevo nivel de recursión (es decir, verá un nuevo nivel **directory | file | directory**). Con el conector de copia total se usan automáticamente todos los niveles de recursión actuales en la instancia XML y no será necesario expandir los niveles de recursión manualmente.

7.3 Importar funciones XSLT 1.0/2.0 personales

Puede ampliar las bibliotecas de funciones XSLT 1.0 y 2.0 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. En el menú **Herramientas** haga clic en **Opciones** (o haga clic en **Agregar o quitar bibliotecas** en la parte inferior de la ventana Bibliotecas).
2. Haga clic en el botón **Agregar** y navegue hasta el archivo .xsl o .xslt.

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 seleccionado es XSLT (*véase Seleccionar un lenguaje de transformación*).

Notas:

- 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.

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
Ejemplo: sumar valores de nodos
Implementación del motor XSLT 1.0
Implementación del motor XSLT 2.0

7.3.1 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 **<Documentos>\Altova\MapForce2018\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.

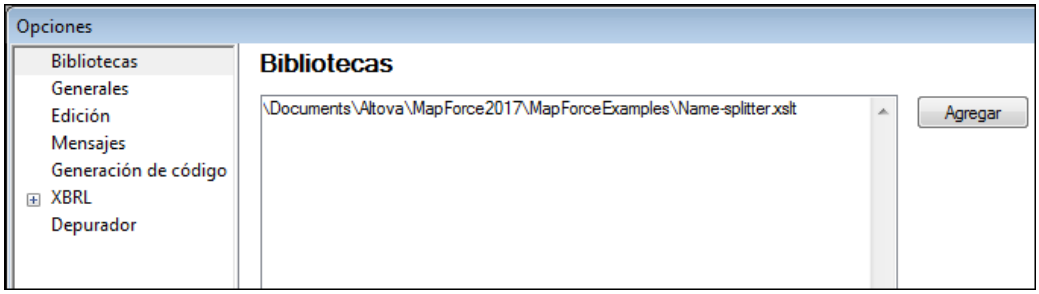
```
2  <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform
3    <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>
4
5    <xsl:template match="*">
6      <xsl:for-each select=".">
7      <xsl:call-template name="tokenize">
8      <xsl:with-param name="string" select="."/>
9      </xsl:call-template>
10     </xsl:for-each>
11   </xsl:template>
12
13   <xsl:template name="tokenize">
14     <xsl:param name="string" select="."/>
15     <xsl:variable name="caps" select="translate($string, '-abcdefghijklmnopqrstuv
16     <xsl:variable name="capscount" select="string-length($caps)"/>
17     <xsl:variable name="token">
```

- **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.

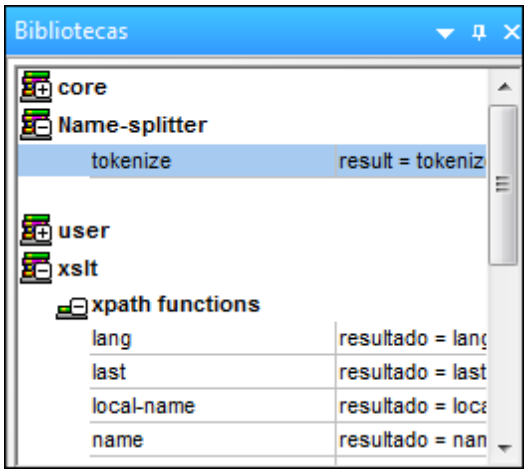
Para agregar una función XSLT personal:

1. Seleccione el lenguaje de transformación XSLT (*véase Seleccionar un lenguaje de transformación*).
2. Haga clic en el botón **Agregar o quitar bibliotecas** situado en la parte inferior de la ventana Bibliotecas (también puede hacer clic en **Herramientas | Opciones** y seleccionar **Bibliotecas**).
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**).

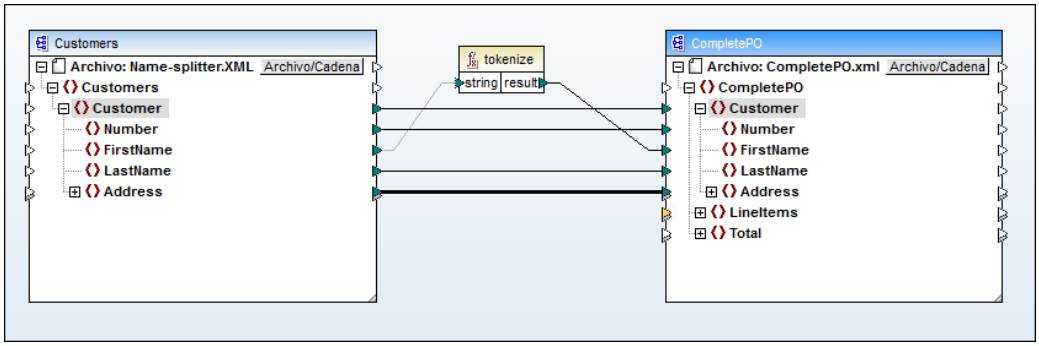


4. Haga clic en **Aceptar**. El nombre del archivo XSLT aparece en la ventana Bibliotecas junto con las funciones definidas como plantillas con nombre (en este ejemplo es **Name-splitter** con la función **`tokenize`**).



Para usar la función XSLT en la asignación de datos:

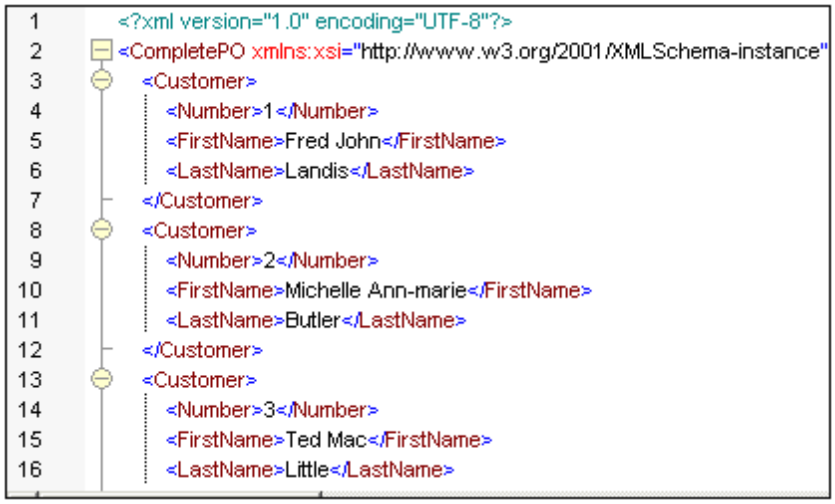
1. Arrastre la función **`tokenize`** hasta el área de asignación y cree las asignaciones de datos que aparecen en esta imagen:



2. Haga clic en el panel *XSLT* para ver el código XSLT de salida.

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**.

7.3.2 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\MapForce2018\MapForceExamples\Tutorial**:

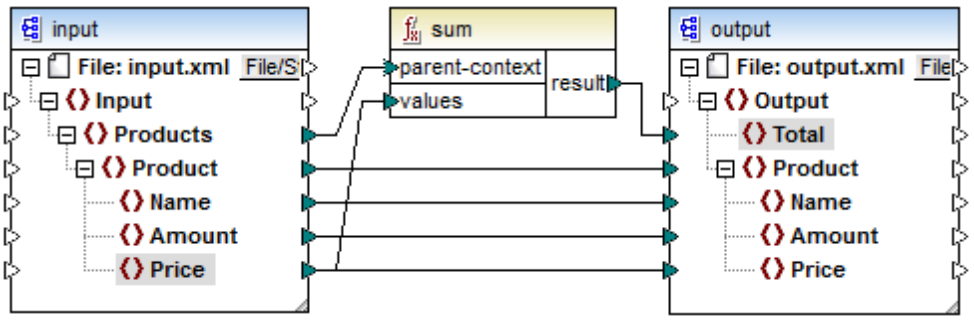
- **Summing-nodes.mfd:** el archivo de asignación de datos
- **input.xml:** el archivo XML de origen
- **input.xsd:** el esquema XML de origen
- **output.xsd:** el esquema XML de destino
- **Summing-nodes.xslt:** una hoja de estilos XSLT personal que contiene una plantilla con nombre que sumará los nodos.

Hay dos maneras de conseguir nuestro objetivo:

- con ayuda de la función de agregado `sum` 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).

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>
```

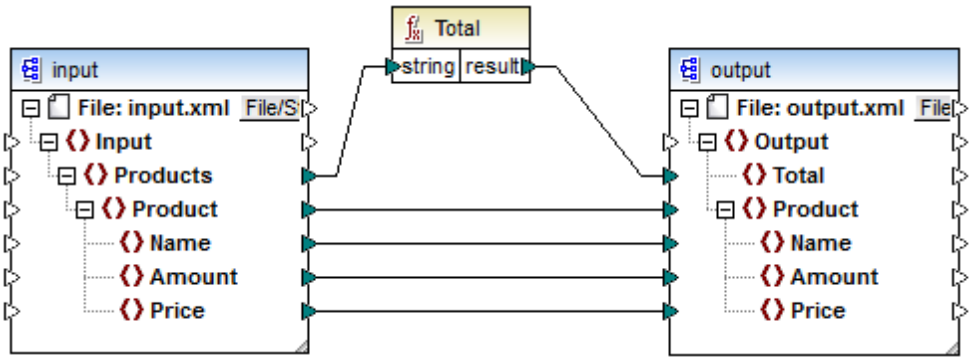
A continuación puede ver una hoja de estilos XSLT personal que usa la plantilla con nombre "Total" y un solo parámetro `string`. La plantilla recorre el archivo XML de entrada y suma todos los valores que obtiene la expresión `/Product/Price`.

```
<?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. Seleccione el lenguaje de transformación XSLT.
2. En la ventana Bibliotecas haga clic en el botón **Agregar o quitar bibliotecas**.
3. En el cuadro de diálogo "Opciones" haga clic en el botón **Agregar**.
4. Navegue hasta el archivo **<Documentos>\Altova\MapForce2018\MapForceExamples\Tutorial\Summing-nodes.xslt**.
5. 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>
```

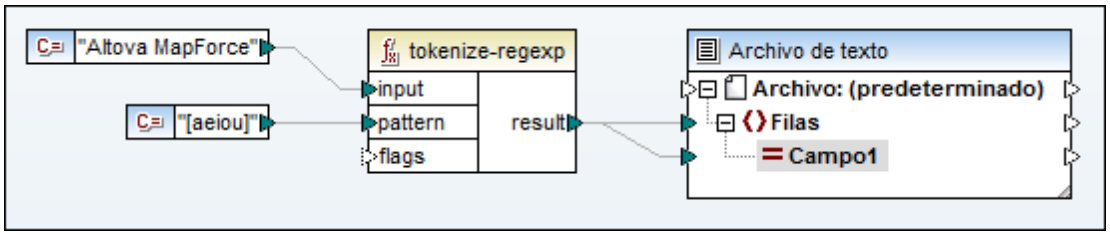
7.4 Expresiones regulares

MapForce admite el uso de expresiones regulares en el parámetro **pattern** de la función **tokenize-regexp** para buscar cadenas concretas.

La sintaxis y la semántica de las expresiones regulares para XLST y XQuery es la definida en http://www.w3.org/TR/xmlschema-2/. Tenga en cuenta que la sintaxis de las expresiones regulares varía de un lenguaje de programación a otro.

Terminología:

input	la cadena en la que trabaja la expresión regular
pattern	la expresión regular
flags	parámetro opcional para definir cómo se debe interpretar la expresión regular
result	el resultado de la función



La función **tokenize-regexp** devuelve una secuencia de cadenas. La conexión con elemento **Filas** crea una fila por cada elemento de la secuencia.

Sintaxis de las expresiones regulares

Literales (un solo carácter):
por ejemplo, la letra "a" es la expresión regular más básica. Busca la primera aparición del carácter "a" de la cadena.

Clases de caracteres []

Conjunto de caracteres entre corchetes.

Solamente se busca **uno** de los caracteres que aparecen entre corchetes.

pattern **[aeiou]**
Busca una vocal en minúsculas.

pattern **[mj]ust**
Busca *must* o *just.*

No olvide que el parámetro **pattern** distingue entre mayúsculas y minúsculas. Por ejemplo, **a** no encontrará **A**.

Intervalos de caracteres **[a-z]**
Crea el intervalo comprendido entre dos caracteres. Solamente se encontrará uno de los

caracteres en cada búsqueda.

pattern **[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.

pattern **[^a-z]**
Busca caracteres que no estén en la clase de caracteres, incluidas las líneas nuevas.

Metacaracteres "."

Este metacarácter busca un solo carácter, sea cual sea (excepto líneas nuevas)

pattern **.**
Busca un solo carácter, sea cual sea.

Cuantificadores ? + * {}

Los cuantificadores definen cuántas veces debe aparecer un componente de la expresión regular dentro de la cadena **`input`** para que la búsqueda obtenga resultados.

?
cero o una — busca cero o una coincidencia de la cadena precedente (la cadena es opcional)

+
una o más — busca una o más coincidencias de la cadena precedente

*****
cero o más — busca cero o más coincidencias de la cadena precendente

{}
mínimo/máximo de repeticiones — número de repeticiones de la cadena o trozo

p. ej. mo{1,3} encuentra `mo`, `moo`, `mooo`.

()
subpatrones
los paréntesis se usan para agrupar partes de la expresión regular.

|
Alternancia/OR
permite probar de izquierda a derecha las subexpresiones
(horse|make) sense encuentra `horse sense` o `make sense`

Parámetro 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*.

El metacarácter "**.**" encuentra un carácter, sea el que sea. 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`**.

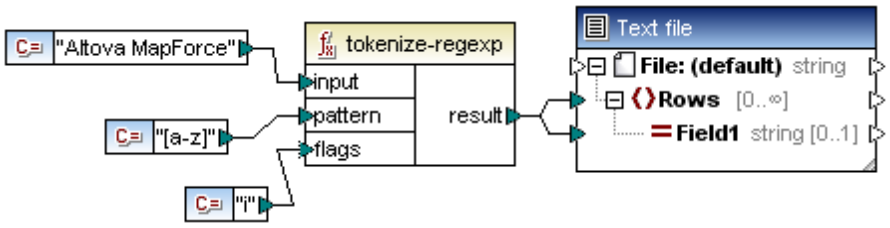
m
Si está presente, el proceso de búsqueda opera en el modo multilínea.

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.
La expresión regular `[a-z]` más la marca **i** encontrará todas las letras de la a-z (en minúsculas) y entre A-Z (en mayúsculas).



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`.

Excepción:
Los caracteres de espacio en blanco sin expresiones de clases de caracteres no se eliminan (por ejemplo, `[#x20]`).

Nota: cuando se genera código, las características avanzadas de la sintaxis regex pueden variar de unos lenguajes a otros. Para más información consulte la documentación de su lenguaje.

7.5 Referencia de la biblioteca de funciones

En esta sección se describen todas las funciones integradas de MapForce que aparecen en la ventana Bibliotecas.

Dependiendo del lenguaje de transformación que esté seleccionado, la ventana Bibliotecas mostrará unas funciones u otras (*véase* *Seleccionar un lenguaje de transformación*).

Limitaciones de las funciones XPath 2.0: varias funciones XPath 2.0 relacionadas con secuencias no están disponibles.

7.5.1 core | aggregate functions (agregado)

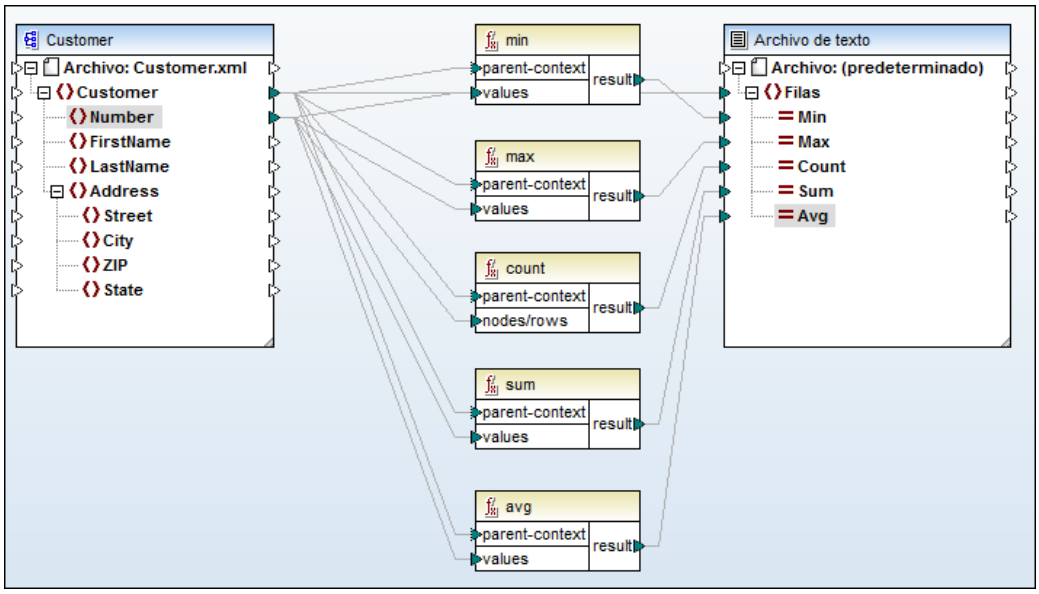
Las funciones de agregado de la biblioteca **core | aggregate** realizan operaciones en un conjunto o secuencia de valores de entrada. Los datos de entrada para las funciones **`min`**, **`max`**, **`sum`** y **`avg`** se convierten al tipo de datos `decimal` para poder procesarlos.

- Los valores de entrada deben conectarse al parámetro **`values`** de la función.
- Puede conectar un nodo de contexto (elemento) al parámetro **`parent-context`** para reemplazar el contexto predeterminado desde el que se toma la secuencia de entrada. El parámetro **`parent-context`** es un parámetro opcional.
- El parámetro **`result`** de la función debe conectarse al elemento de destino.

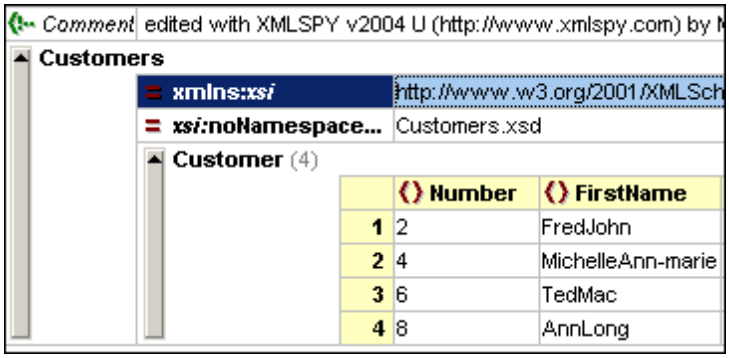
La asignación que aparece a continuación es la del archivo `Aggregates.mfd`, guardado en la carpeta `...\Tutorial` y explica cómo utilizar este tipo de funciones.

Las funciones de agregado tienen dos entradas:

- la entrada **`values`** (**`nodes/rows`**) se conecta al elemento del componente de origen que suministra los datos, en este caso el elemento **`Number`**.
- la entrada **`parent-context`** se conecta al elemento en el que se desea iterar (es decir, el contexto. En este caso se quiere iterar en todos los elementos **`Customer`**). Este parámetro es opcional.

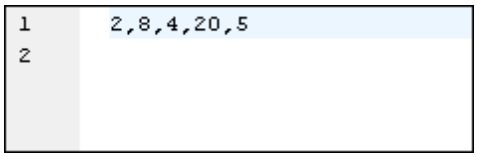


En este caso la instancia de entrada es un archivo XML que contiene estos datos:



- Los datos de origen suministrados a la entrada **values** es la secuencia de números 2,4,6,8.
- El componente de salida en este caso es un simple archivo de texto.

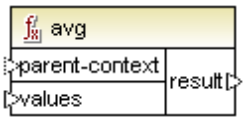
  La asignación anterior genera estos resultados:



min=2, max=8, count=4, sum=20 y avg=5.

7.5.1.1 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. Esta función no está disponible en XSLT1.

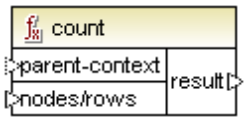


Argumento	Descripción
parent-context	Argumento opcional. Suministra el contexto primario (*véase Reemplazar el contexto de asignación*).
values	Este argumento debe estar conectado a un elemento de entrada que suministra los datos propiamente dichos. Recuerde que el valor de argumento dado debe ser numérico.

Para ver un ejemplo de uso consulte el diseño de asignación de datos **GroupTemperaturesByYear.mfd** del directorio **<Documentos>\Altova\MapForce2018\MapForceExamples**.

7.5.1.2 count

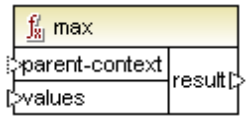
Devuelve el número de elementos que componen la secuencia de entrada. El recuento de un conjunto vacío es 0. Esta función no es totalmente compatible con XSLT1.



Argumento	Descripción
parent-context	Argumento opcional. Suministra el contexto primario (*véase Reemplazar el contexto de asignación*).
nodes/rows	Este argumento debe estar conectado al elemento de origen que se debe contar.

7.5.1.3 max

Devuelve el valor máximo de todos los valores numéricos de la secuencia de entrada. Tenga en cuenta que esta función devuelve un conjunto vacío si el argumento **strings** es un conjunto vacío. Esta función no está disponible en XSLT1.

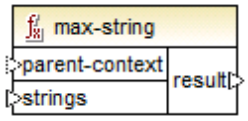


Argumento	Descripción
parent-context	Argumento opcional. Suministra el contexto primario (*véase Reemplazar el contexto de asignación*).
values	Este argumento debe estar conectado a un elemento de entrada que suministra los datos propiamente dichos. Recuerde que el valor de argumento dado debe ser numérico. Para obtener el máximo de una secuencia de cadenas utilice la función **`max-string`**.

Para ver un ejemplo de uso consulte el diseño de asignación de datos **GroupTemperaturesByYear.mfd** del directorio **<Documentos>\Altova\MapForce2018\MapForceExamples**.

7.5.1.4 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". Esta función no está disponible en XSLT1.

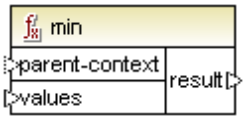


Argumento	Descripción
parent-context	Argumento opcional. Suministra el contexto primario (*véase Reemplazar el contexto de asignación*).
strings	Este argumento debe estar conectado a un elemento de entrada que suministra los datos propiamente dichos. El valor de argumento dado debe ser una secuencia (cero o varios) de `xs:string`.

Tenga en cuenta que la función devuelve un conjunto vacío si el argumento **strings** es un conjunto vacío.

7.5.1.5 min

Devuelve el valor mínimo de todos los valores numéricos de la secuencia de entrada. El mínimo de un conjunto vacío es un conjunto vacío. Esta función no está disponible en XSLT1.



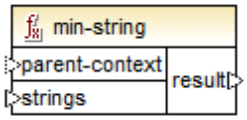
Argumento	Descripción
parent-context	Argumento opcional. Suministra el contexto primario (*véase Reemplazar el*

Argumento	Descripción
	contexto de asignación).
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 mínimo de una secuencia de cadenas utilice la función **`min-string`**.

Para ver un ejemplo de uso consulte el diseño de asignación de datos **GroupTemperaturesByYear.mfd** del directorio **<Documentos>\Altova\MapForce2018\MapForceExamples**.

7.5.1.6 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". Esta función no está disponible en XSLT1.

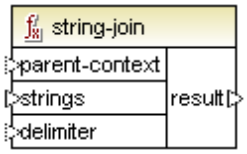


Argumento	Descripción
parent-context	Argumento opcional. Suministra el contexto primario (*véase Reemplazar el contexto de asignación*).
strings	Este argumento debe estar conectado a un elemento de entrada que suministra los datos propiamente dichos. El valor de argumento dado debe ser una secuencia (cero o varios) de `xs:string`.

Tenga en cuenta que la función devuelve un conjunto vacío si el argumento **strings** es un conjunto vacío.

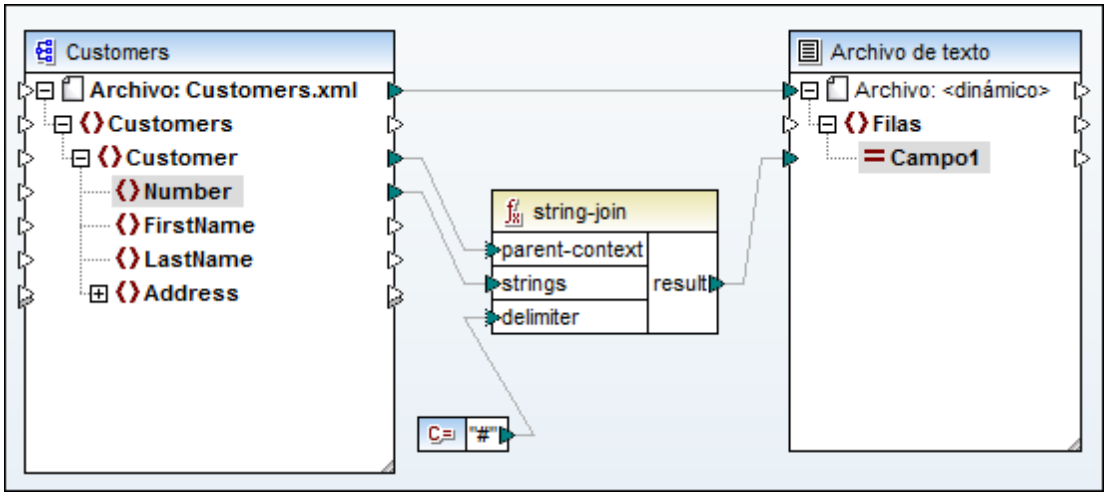
7.5.1.7 string-join

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. Esta función no está disponible en XSLT1.



El ejemplo de la siguiente imagen muestra cuatro números: 2, 4, 6 y 8. La constante ofrece el carácter # como delimitador.

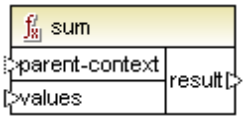
Resultado = 2#4#6#8



Si no especifica un delimitador (**delimiter**), el valor de argumento predeterminado es una cadena vacía (es decir, ningún delimitador). Resultado = 2468.

7.5.1.8 sum

Devuelve la suma aritmética de todos los valores de la secuencia de entrada. La suma de un conjunto vacío es 0.



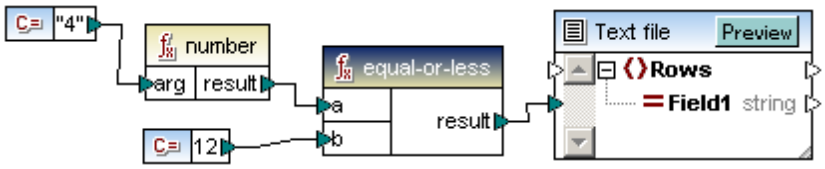
Argumento	Descripción
parent-context	Argumento opcional. Suministra el contexto primario (*véase Reemplazar el contexto de asignación*).
values	Este argumento debe estar conectado a un elemento de entrada que suministra los datos propiamente dichos. Recuerde que el valor de argumento dado debe ser numérico.

También puede consultar el apartado Ejemplo: sumar valores de nodo.

7.5.2 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.



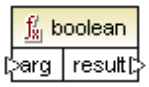
En el ejemplo de la imagen la primera constante es de tipo string y contiene la cadena "4". La segunda constante contiene la constante numérica 12. Para poder comparar estos dos valores de forma explícita los tipos deben coincidir.

Si añadimos una función **number** a la primera constante, la constante de cadena se convierte en el valor numérico 4. El resultado de la comparación será "true".

Si no se usara la función **number** (es decir, si 4 estuviera conectado al parámetro **a** directamente), se produciría una comparación de cadenas y su resultado sería "false".

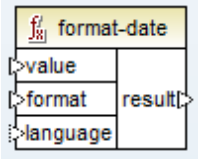
7.5.2.1 *boolean*

Convierte un valor numérico de entrada en un valor booleano (o un valor de cadena en un valor numérico) P. ej. `0` se convierte en `false` o `1` se convierte en `true`, valores que después se pueden usar en funciones lógicas (como **`equal`**, **`greater`**, etc.), en filtros o en condiciones **`if-Else`**.



7.5.2.2 *format-date*

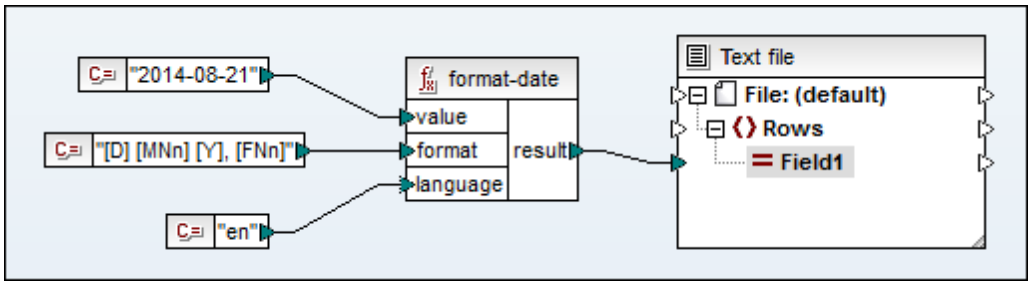
Convierte un valor de entrada `xs:date` en un valor de cadena y le aplica el formato indicado por las opciones especificadas.



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 función **`format-dateTime`**.
language	Argumento opcional que devuelve el nombre del mes y el día de la semana en el idioma seleccionado. Valores válidos:

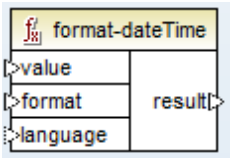
Argumento	Descripción
	en (predet.) inglés **es** español **de** alemán **ja** japonés

En la imagen siguiente, por ejemplo, el resultado es "21 August 2014, Thursday". Para traducir este valor al idioma español, defina el valor `es` para el argumento `language`.



7.5.2.3 *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`.



Argumento	Descripción
value	El valor `dateTime` al que se debe aplicar formato.
format	Cadena de formato que identifica el formato que se debe aplicar a **value**.
language	Argumento opcional que devuelve el nombre del mes y el día de la semana en el idioma seleccionado. Valores válidos: **en (predet.)** inglés **es** español **de** alemán **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 de los componentes*).

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
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)]

Note: los modificadores N, n y Nn solamente son compatibles con estos componentes: M, d, D.

El modificador del ancho, si existe, se introduce con una coma y toma este formato:

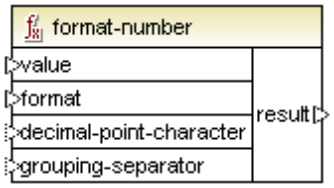
```
, min-width ("-" max-width)?
```

La tabla siguiente muestra algunos ejemplos de formato de valores `xs:dateTime`, obtenido con ayda 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]`	3/11/2003
2003-11-03T00:00:00	`[Y]-[M,2]-[D,2]`	2003-11-03
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

7.5.2.4 format-number

Convierte un número en una cadena. La función está disponible para XSLT 1.0, XSLT 2.0, Java, C#, C++ y el motor de ejecución integrado.



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. Este argumento se usa igual que el argumento `format` de la función **`format-dateTime`**.
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 (`.`).
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 (es decir, **`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.

Format:

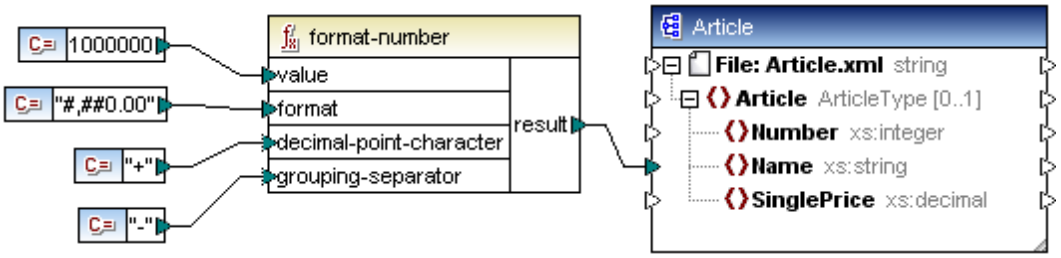
```
format := subformat (;subformat)?
  subformat := (prefix)? integer (.fraction)? (suffix)?
  prefix := any characters except special characters
  suffix := any characters except special characters
  integer := (#)* (0)* ( allowing ',' to appear)
  fraction := (0)* (#)* (allowing ',' to appear)
```

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

Carácter especial	Predeterminado	Descripción
		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
separador de grupos	,	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. No obstante, se pueden cambiar en el resultado formateado asignando constantes a estos nodos.



El resultado de la función **`format-number`** anterior es:

- El carácter de punto decimal se cambió por +.
- El separador de grupos se cambió por -.

```
<Article xsi:noNamespaceSchemaLocation='
  <Name>1-000-000+00</Name>
</Article>
```

Redondeo

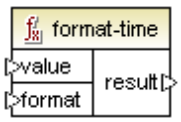
El método de redondeo utilizado para esta función es al alza (p. ej. redondea al alza si la fracción es mayor o igual a 0.5 y redondea a la baja si la fracción es menor que 0.5). Este método de redondeo solamente afecta al código generado y si la opción seleccionada es el motor de ejecución integrado.

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.

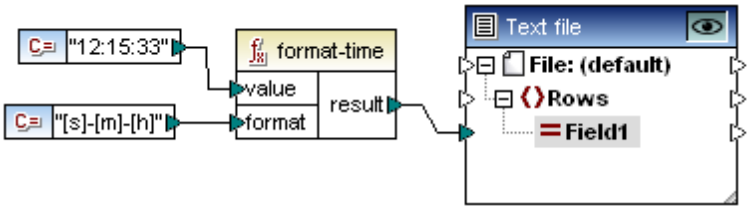
Número	Cadena de formato	Resultado
1234.5	#,##0.00	1,234.50
123.456	#,##0.00	123.46
1000000	#,##0.00	1,000,000.00
-59	#,##0.00	-59.00
1234	###0.0###	1234.0
1234.5	###0.0###	1234.5
.00025	###0.0###	0.0003
.00035	###0.0###	0.0004
0.25	#00%	25%
0.736	#00%	74%
1	#00%	100%
-42	#00%	-4200%
-3.12	#.00;(#.00)	(3.12)
-3.12	#.00;#.00CR	3.12CR

7.5.2.5 *format-time*

Convierte un valor de entrada `xs:time` en un valor de cadena. El argumento **format** se usa igual que el argumento `format` de la función **`format-dateTime`**.



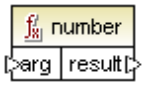
Ejemplo:



Resultado: 33-15-12

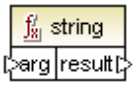
7.5.2.6 *number*

Convierte una cadena de entrada en un número. También convierte una entrada booleana en un número.



7.5.2.7 string

Convierte un valor de entrada en una cadena. La función también se puede usar para recuperar el contenido de texto de un nodo.



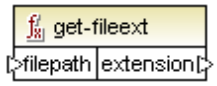
Si el nodo de entrada es un tipo complejo XML, todos los descendientes también se generan como una sola cadena.

7.5.3 core | file path functions (ruta de archivos)

Las funciones de ruta de acceso de la biblioteca **core | 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.

7.5.3.1 get-fileext

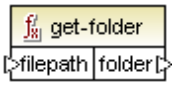
Devuelve la extensión de la ruta de archivo, incluido el carácter "."



P. ej. `c:\data\Sample.mfd` devuelve "`.mfd`"

7.5.3.2 get-folder

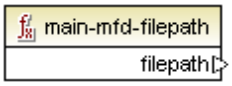
Devuelve el nombre de la carpeta de la ruta de archivo, incluida la barra diagonal final o la barra diagonal inversa final.



P. ej. `c:/data/Sample.mfd` devuelve `c:/data/`

7.5.3.3 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.



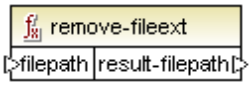
7.5.3.4 mfd-filepath

Si se le llama desde la asignación principal, la función devuelve el mismo resultado que la función **`main-mfd-filepath`**, es decir, la ruta completa del archivo `mfd` que contiene la asignación principal. Si el archivo `mfd` 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.

7.5.3.5 remove-fileext

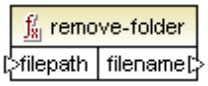
Borra la extensión de la ruta de archivo, incluido el carácter "."



P. ej. `c:/data/Sample.mfd` devuelve `c:/data/Sample`.

7.5.3.6 remove-folder

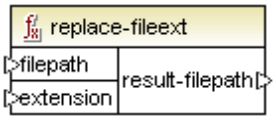
Borra el directorio de la ruta de acceso, incluida la barra diagonal final o la barra diagonal inversa final.



P. ej. `c:/data/Sample.mfd` devuelve `Sample.mfd`.

7.5.3.7 replace-fileext

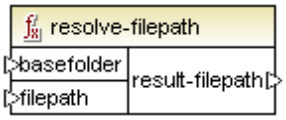
Reemplaza la extensión de la ruta de acceso dada por el parámetro **`filepath`** con la extensión dada por el parámetro **`extension`**.



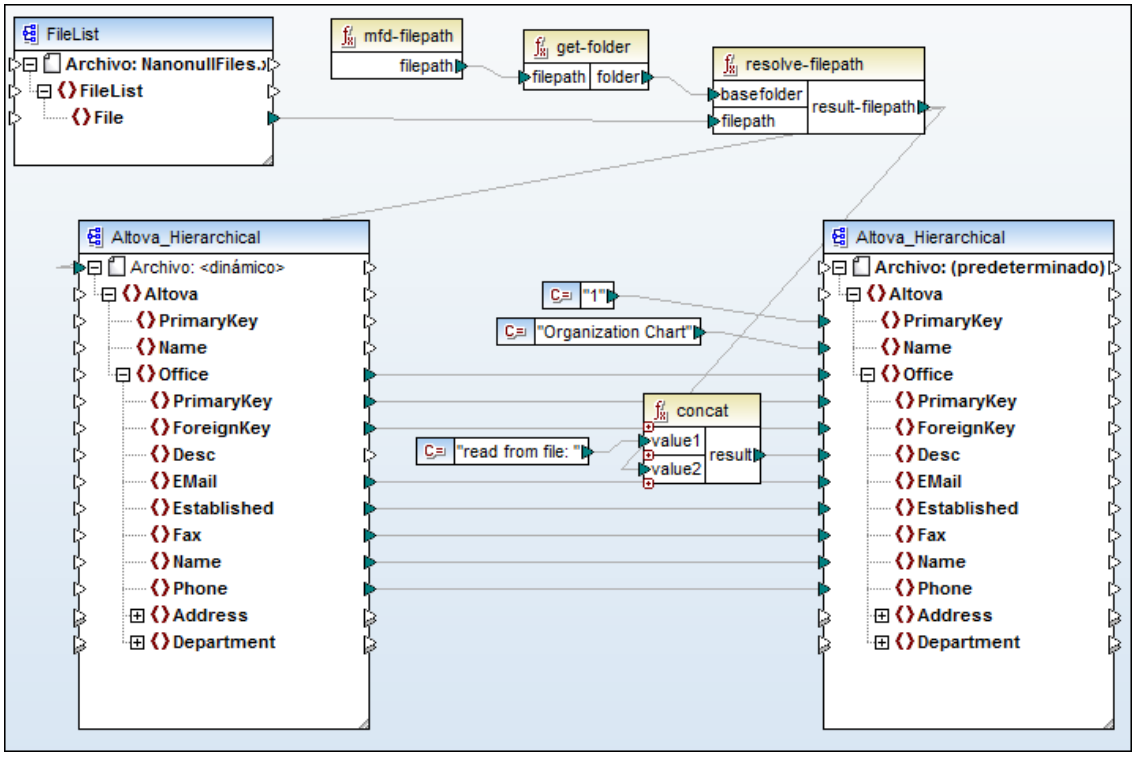
P. ej. si el parámetro **filepath** es la ruta de acceso `c:/data/Sample.mfd` y `.mfp` es la extensión del parámetro **extension**, la función devuelve `c:/data/Sample.mfp`.

7.5.3.8 resolve-filepath

Convierte una ruta de acceso relativa en una carpeta base relativa o absoluta. La función admite "." (directorio actual) y ".." (directorio primario).



Para ver un ejemplo abra el archivo de asignación **MergeMultipleFiles_List.mfd** , disponible en la carpeta ...\MapForceExamples.



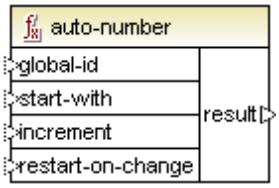
7.5.4 core | generator functions (generador)

La biblioteca de funciones **core / generator** incluye funciones que generan valores.

7.5.4.1 auto-number

La función **auto-number** genera enteros en los nodos de destino de un componente, en base a los parámetros definidos. El resultado es un valor que empieza en el parámetro **start_with** y se

incrementa con el valor de **increment**. Los valores predeterminados son **start-with=1** e **increase=1**. Ambos parámetros pueden ser negativos.

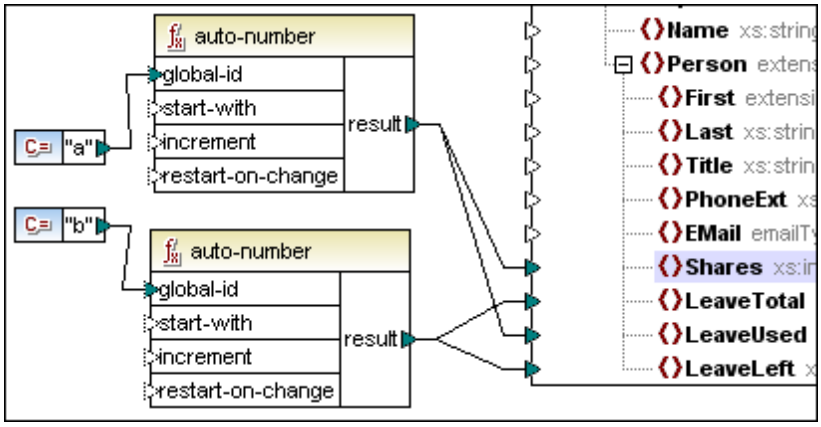


Compruebe que el conector del parámetro **result** (de la función **auto-number**) está conectado directamente al nodo de destino. Recuerde que el código generado no llama a las funciones en ningún orden en particular. MapForce puede copiar en caché los resultados calculados para volver a utilizaros o evaluar las expresiones en cualquier orden. Por tanto, recomendamos encarecidamente que utilice con cuidado la función **auto-number**.

Parámetro global-id

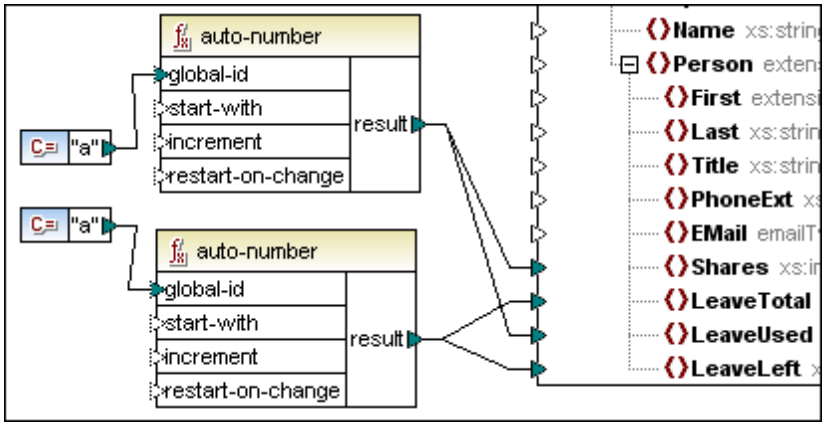
Sirve para sincronizar la secuencia numérica resultante de dos funciones **auto-number** distintas conectadas al mismo componente de destino. Si las dos funciones **auto-number** no tienen el mismo **global-id**, cada función incrementa los elementos de destino por separado. En el ejemplo siguiente, las funciones tienen parámetros **global-id** diferentes (**a** y **b**).

El resultado de la asignación es `1,1,2,2`. La primera función (con **global-id a**) aporta el primer 1 y la segunda (con **global-id b**) aporta el segundo 1.



Si ambas funciones tienen **el mismo global-id**, digamos **a**, entonces ambas funciones conocen el estado actual (o valor real) de la otra y ambos números están sincronizados (*imagen siguiente*).

En este caso el resultado de la asignación es `1, 2, 3, 4`. La primera función aporta el primer 1 y la segunda función ahora aporta un 2.

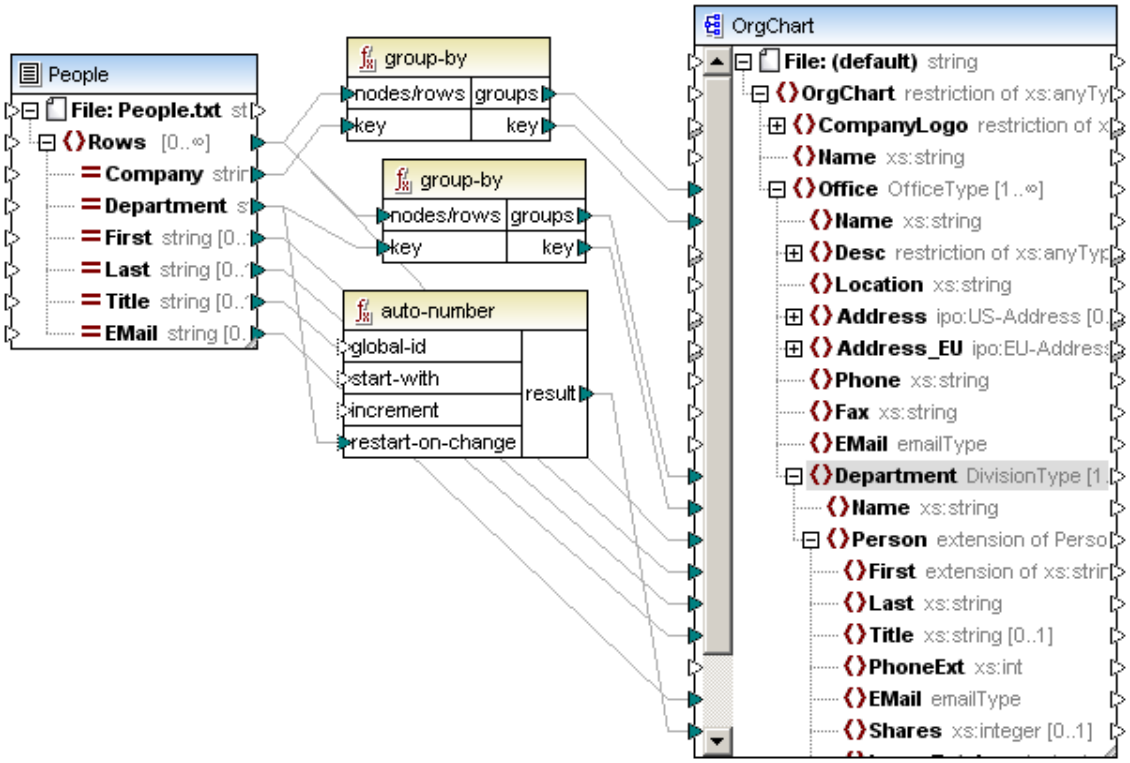


Parámetro start-with
El valor inicial utilizado para comenzar la secuencia de numeración automática. Valor predeterminado = 1.

Parámetro increment
El incremento que se debe aplicar a la secuencia de numeración automática. Valor predeterminado = 1.

Parámetro restart-on-change
Restablece el contador de numeración automática al parámetro **`start-with`** cuando cambia el **contenido** de los elementos conectados.

En el ejemplo siguiente, **`start-with`** e **`increment`** tienen el valor predeterminado 1. En cuanto cambia el **contenido** de **`Department`** (es decir, cambia el nombre del departamento), el contador se restablece y la secuencia de cada departamento nuevo empieza por 1.



7.5.5 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, a un subconjunto del componente de destino.

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 entero 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".

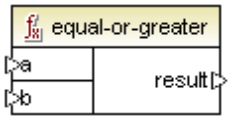
Nota: las funciones lógicas no se pueden usar para comprobar la existencia de valores null. Si se da un valor null como argumento de una función lógica, la función devuelve un valor null. Para más información consulte el apartado Valores Nil y Nillable.

7.5.5.1 *equal*

El resultado es **`true`** si **`a=b`**. De lo contrario, el resultado es **`false`**.

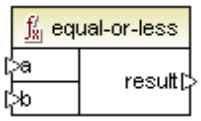
7.5.5.2 *equal-or-greater*

El resultado es **`true`** si **`a`** es igual/mayor que **`b`**. De lo contrario, el resultado es **`false`**.



7.5.5.3 *equal-or-less*

El resultado es **`true`** si **`a`** es igual o menor que **`b`**. De lo contrario, el resultado es **`false`**.



7.5.5.4 *greater*

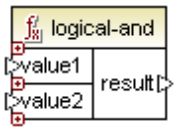
El resultado es **`true`** si **`a`** es mayor que **`b`**. De lo contrario, el resultado es **`false`**.

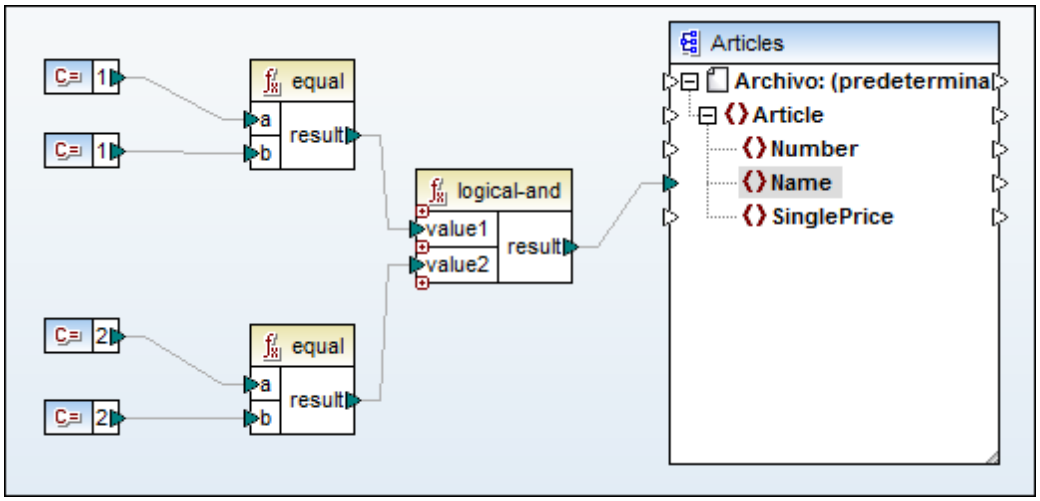
7.5.5.5 *less*

El resultado es **`true`** si **`a`** es menor que **`b`**. De lo contrario, el resultado es **`false`**.

7.5.5.6 *logical-and*

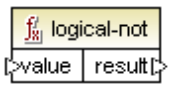
Si tanto **value1** como **value2** son **true**, el resultado es **true**. Si son diferentes, el resultado es **false**.



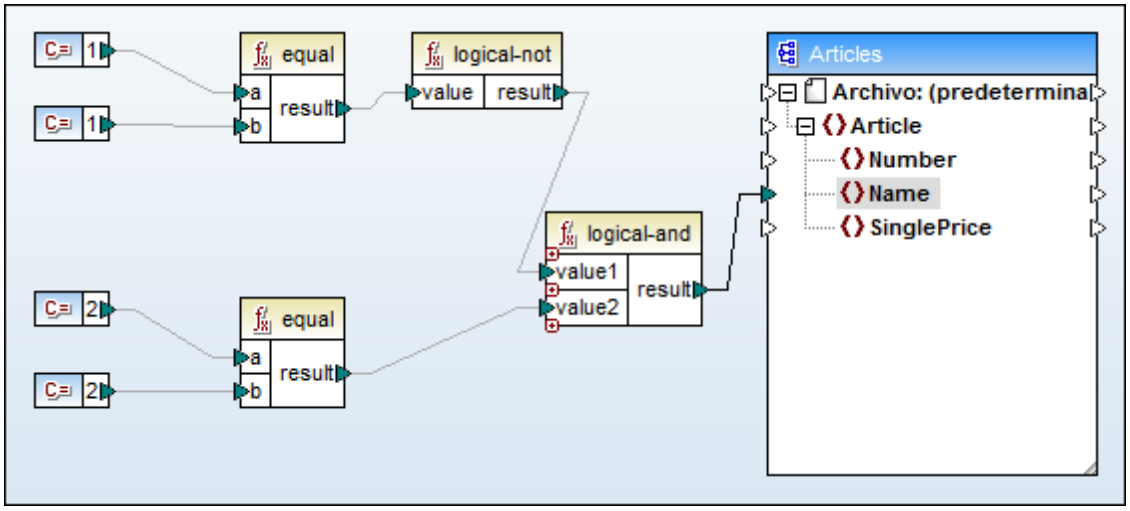


7.5.5.7 *logical-not*

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 **false**, el resultado de la función es **true**.

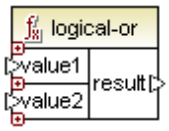


La función **logical-not** que aparece a continuación invierte el resultado de la función **equal**. La función **logical-and** ahora solamente devuelve el valor **true** si los valores booleanos de **value1** y **value2** son diferentes (es decir, **true-false** o **false-true**).



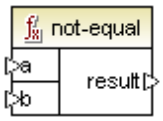
7.5.5.8 *logical-or*

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`**.



7.5.5.9 *not-equal*

El resultado es **`true`** si **`a`** no es igual a **`b`**.

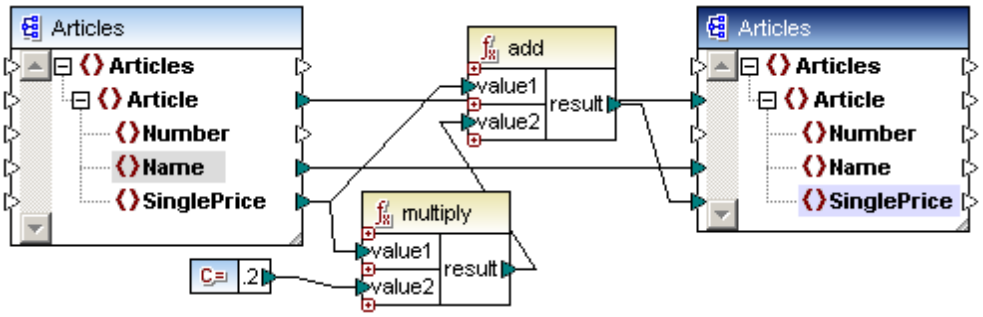


7.5.6 core | math functions (matemáticas)

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`.

Parámetros de entrada = **`value1 | value2`**
Parámetro de salida = **`result`**

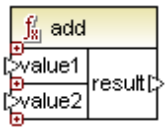
MapForce convierte automáticamente los valores de entrada en decimales para poder procesarlos.



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.

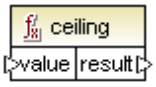
7.5.6.1 add

El resultado es el valor decimal resultante de sumar **value1** a **value2**.



7.5.6.2 ceiling

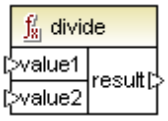
El resultado es el entero más pequeño que es mayor o igual que **value**, es decir, el valor entero mayor más próximo al valor decimal de entrada **value**.



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).

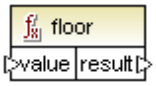
7.5.6.3 divide

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 para definir la precisión del resultado.



7.5.6.4 floor

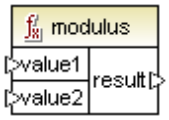
El resultado es el entero mayor que sea menor o igual que **value**, es decir, el valor entero menor más próximo al parámetro decimal **value** de entrada.



P. ej. si el resultado de una función de división es `11.2` y a este resultado le aplicamos la función **floor**, el resultado se convierte en `11` (es decir, el número entero menor más cercano).

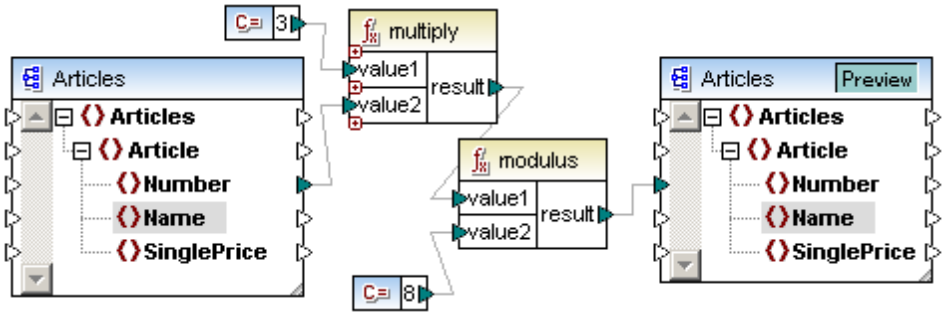
7.5.6.5 modulus

El resultado es el entero restante después de dividir **value1** por **value2**.



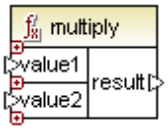
En la asignación que aparece a continuación, los números se multiplicaron por 3 y se pasaron al parámetro **value1** de la función **modulus**. Los valores de entrada ahora son 3, 6, 9 y 12.

Cuando se aplica/utiliza la función **modulus** con **value2** = 8, los enteros restantes son 3, 6, 1 y 4.



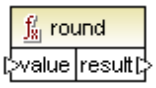
7.5.6.6 multiply

El resultado es el valor decimal resultante de multiplicar **value1** por **value2**.



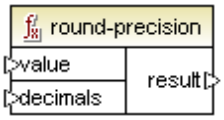
7.5.6.7 round

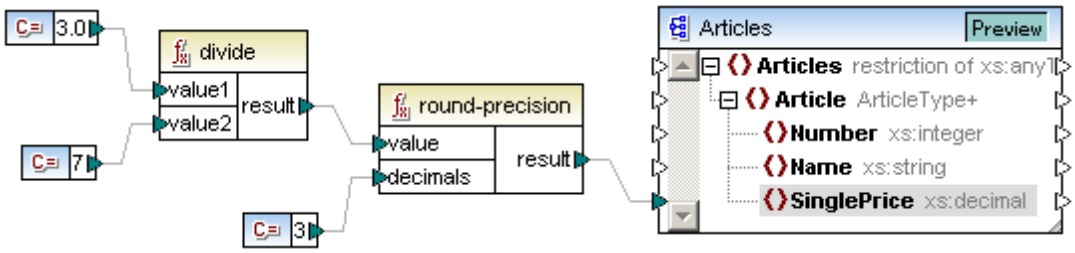
Devuelve el valor redondeado al entero más cercano. Cuando el valor esté justo entre dos enteros, se utilizará el algoritmo *Desempate la mitad alejándose del cero.* Por ejemplo, el valor `10.5` se redondearía hasta `11` y el valor `-10.5` se redondearía hasta `-10`.



7.5.6.8 round-precision

El resultado es el valor decimal del número redondeado a las cifras decimales definidas por el parámetro **`decimals`**.

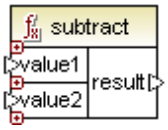




Por ejemplo, en la asignación anterior, el resultado es `0.429`. Si quiere que este resultado aparezca correctamente en el archivo XML, debe estar asignado a un elemento de tipo `xs:decimal`.

7.5.6.9 subtract

El resultado es el valor decimal resultante de restar **`value2`** a **`value1`**.

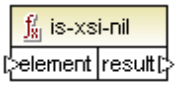


7.5.7 core | node functions (nodo)

Las funciones de la biblioteca **core | node** sirven para obtener acceso a nodos o procesarlos.

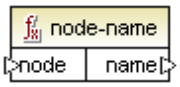
7.5.7.1 *is-xsi-nil*

Devuelve **`true`** (<OrderID>true</OrderID>) si el nodo elemento del componente de origen tiene un atributo `xsi:nil` con el valor `true`.

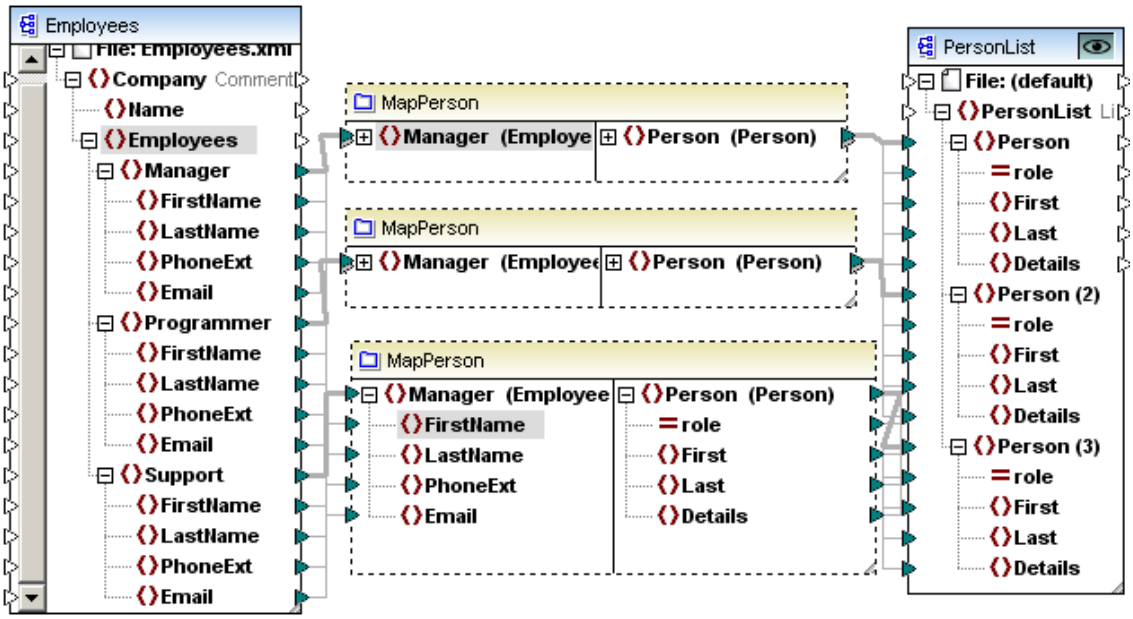


7.5.7.2 *node-name*

Devuelve el QName del nodo conectado, a no ser que sea un nodo XML `text()`. Si es así, devuelve un QName vacío. Esta función solamente funciona con los nodos que tienen nombre. Si XSLT 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.

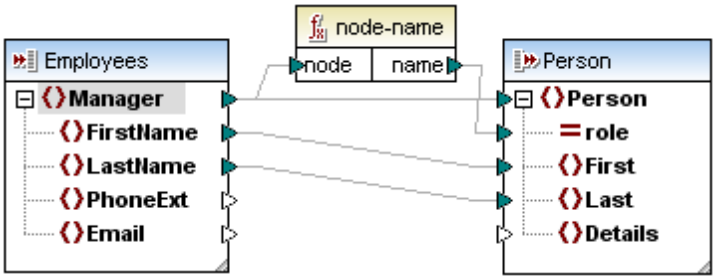


- No puede obtener un nombre de campos/tablas de BD.
- Esta función no es compatible con XBRL ni Excel.
- No se puede obtener un nombre del nodo de entrada **`Archivo`**.
- Los nodos de servicio web se comportan como nodos XML, excepto que:
  - node-name no es compatible con `part`.
  - node-name no es compatible con el nodo raíz (de entrada o salida).



La función definida por el usuario **`MapPerson`** usa la función **`node-name`** para recuperar el nombre del nodo de entrada y colocarlo en el atributo `role`. El nodo raíz del esquema `Employees.xsd`, en

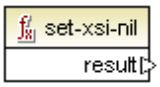
la función definida por el usuario, es **Manager**.



Manager recibe sus datos de **fuera** de la función definida por el usuario y su contenido puede ser `Manager`, `Programmer` o `Support`. Estos son los datos que se pasan al atributo `role` del componente de destino **PersonList**.

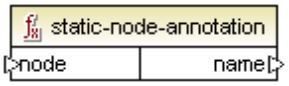
7.5.7.3 *set-xsi-nil*

Establece el nodo de destino en xsi:nil.



7.5.7.4 *static-node-annotation*

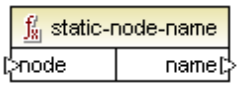
Devuelve una cadena con la anotación del nodo conectado. El parámetro de entrada **node** debe ser (i) un nodo del componente de origen o (ii) una función inline directamente conectada a un parámetro, directamente conectado a su vez a un nodo de la asignación que llama a la función.



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.

7.5.7.5 *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 directamente conectada a un parámetro, directamente conectado a su vez a un nodo de la asignación que llama a la función.

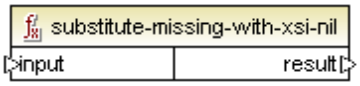


La conexión debe ser directa. No puede pasar por un filtro ni por una función definida por el

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.

7.5.7.6 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.



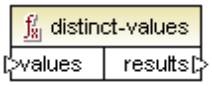
7.5.8 core | sequence functions (secuencia)

Las funciones de secuencia de la biblioteca **core | sequence** sirven para procesar secuencias de entrada y agrupar su contenido. El valor/contenido del parámetro de entrada **key**, asignado a nodos/filas, se utiliza para agrupar la secuencia.

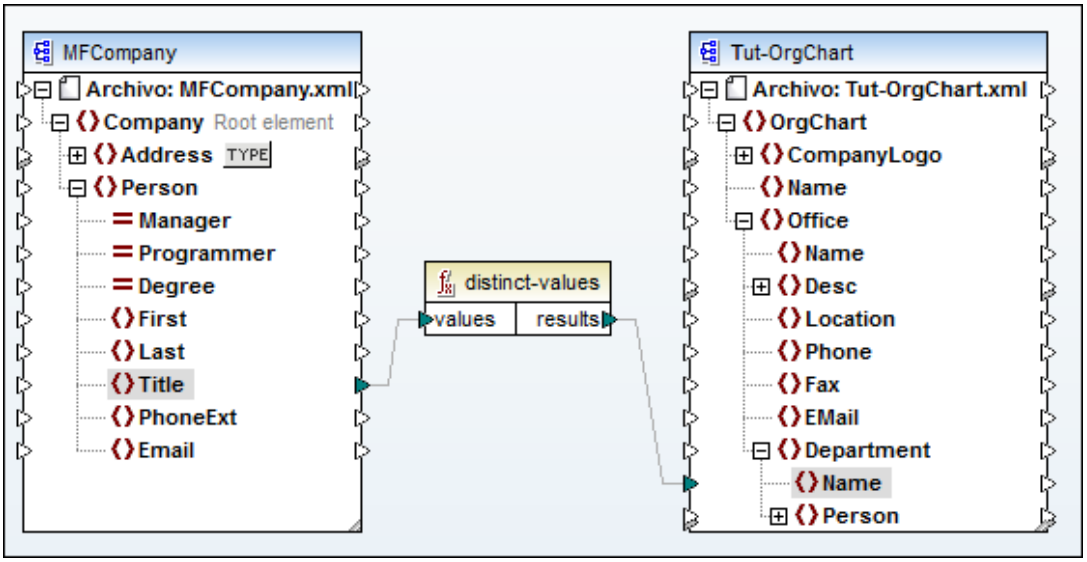
- El parámetro de entrada **key** tiene un tipo de datos arbitrario que se puede convertir en `string` en las funciones de secuencia **group-adjacent** y **group-by**.
- El parámetro de entrada **bool** tiene el tipo de datos **Boolean** en las funciones de secuencia **group-starting-with** y **group-ending-with**.
- El parámetro de salida **key** es la clave del grupo actual.

7.5.8.1 distinct-values

Permite quitar los valores duplicados de una secuencia y asignar los elementos únicos al componente de destino.



En el ejemplo que aparece a continuación se analiza el contenido de los elementos del componente de origen "Title" y cada título que sea único se asigna al elemento Department / Name del componente de destino.

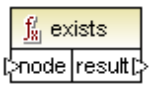


Observe que la secuencia de los elementos Title del componente de origen se conserva cuando se asignan al componente de destino.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <OrgChart xsi:schemaLocation="http://www.xmlspy.com/schemas/orgchart C:/DOCU
3    <Office>
4      <Department>
5        <Name>Office Manager</Name>
6        <Name>Accounts Receivable</Name>
7        <Name>Accounting Manager</Name>
8        <Name>Marketing Manager Europe</Name>
9        <Name>Art Director</Name>
10       <Name>Program Manager</Name>
11       <Name>Software Engineer</Name>
12       <Name>Technical Writer</Name>
13       <Name>IT Manager</Name>
14       <Name>Web Developer</Name>
15       <Name>Support Engineer</Name>
16       <Name>PR &amp;amp; Marketing Manager US</Name>
17     </Department>
18   </Office>
19 </OrgChart>
```

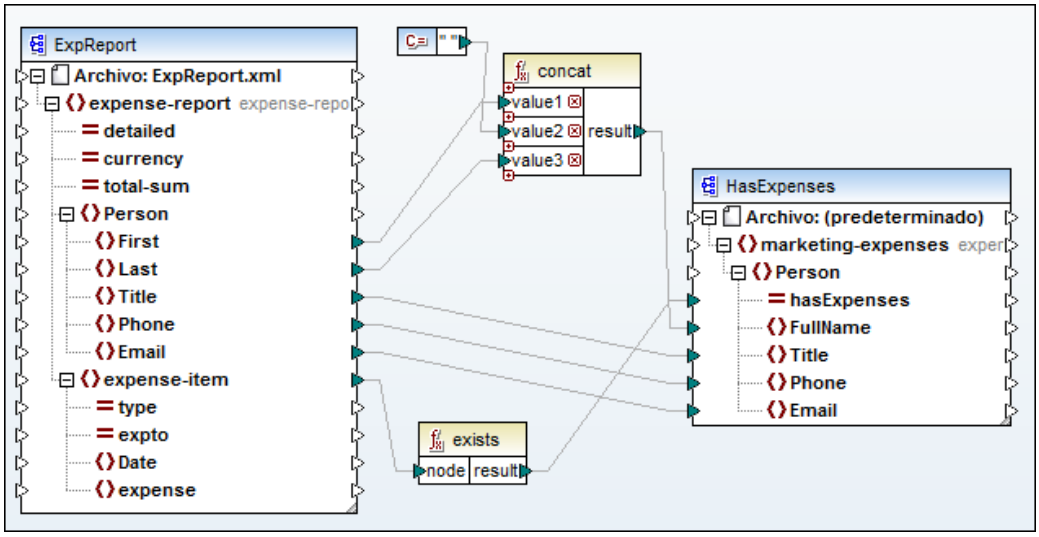
7.5.8.2 exists

Devuelve `true` si el nodo existe. De lo contrario devuelve `false`.



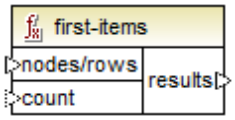
El diseño de asignación de datos **HasMarketingExpenses.mfd** de la carpeta ...\MapForceExamples contiene el ejemplo que aparece en la imagen siguiente.

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.



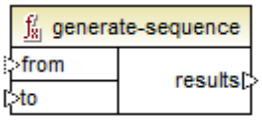
7.5.8.3 *first-items*

Devuelve los X primeros elementos de la secuencia de entrada, siendo X el número que suministra el parámetro **count**. Por ejemplo, si se asigna el valor **3** al parámetro **count** y un nodo primario al parámetro **nodes/rows**, entonces el resultado de la función serán los tres primeros elementos del nodo primario.



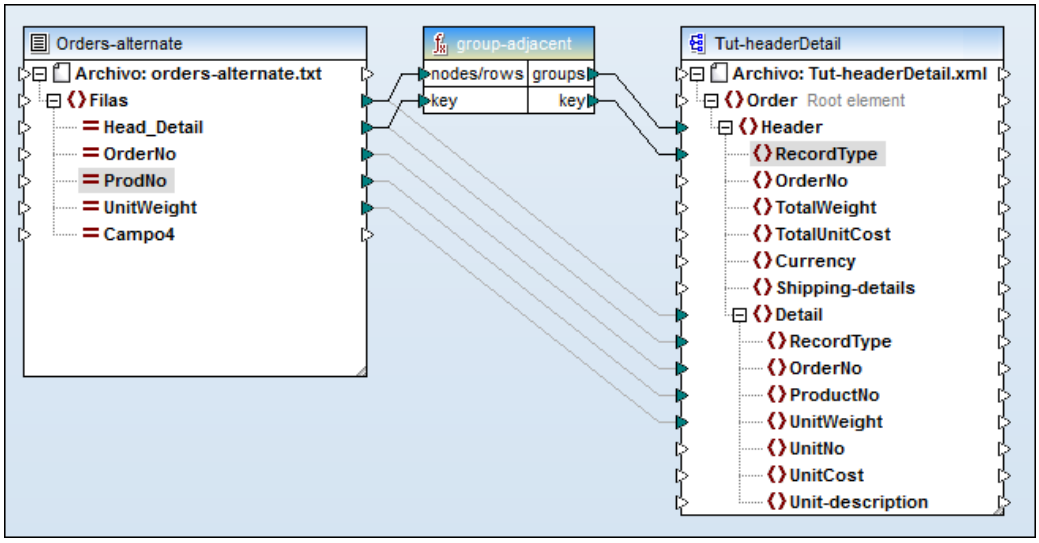
7.5.8.4 *generate-sequence*

Crea una secuencia de enteros usando como límite los parámetros **from** y **to**.



7.5.8.5 *group-adjacent*

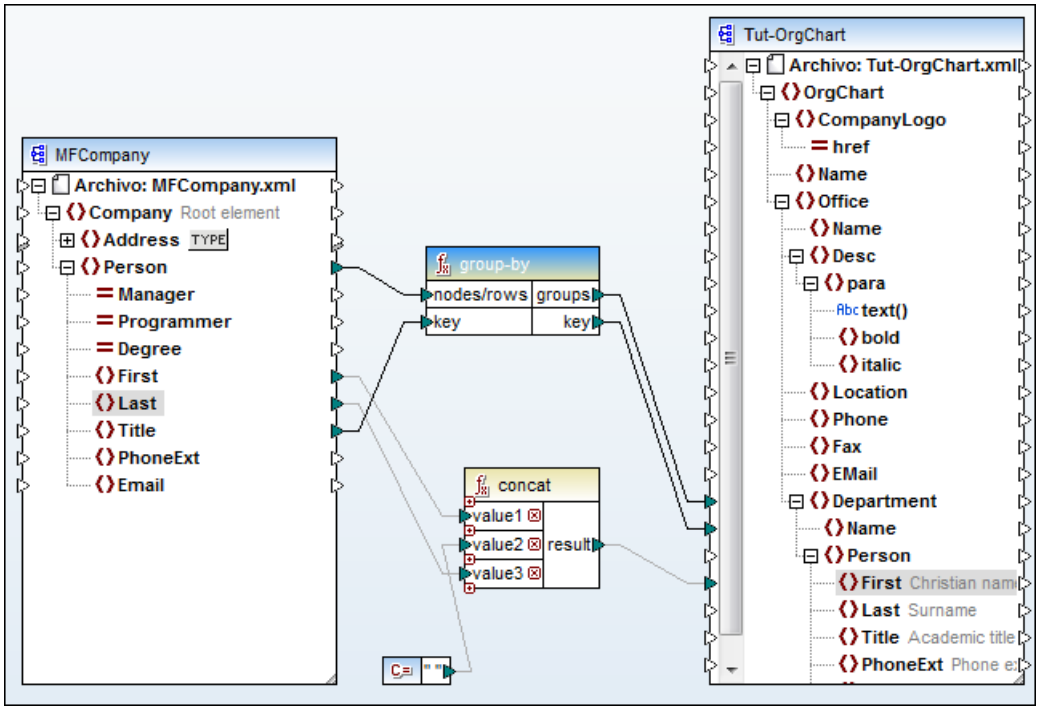
Agrupa la secuencia del parámetro de entrada **nodes/rows** en grupos de elementos adyancentes que comparten la misma clave **key**. Observe que la función **group-adjacent** utiliza el contenido del nodo/elemento como clave de agrupación.



7.5.8.6 group-by

Agrupa la secuencia del parámetro de entrada **nodes/rows** en grupos de elementos no necesariamente adyacentes que comparten la misma clave **key**. Los grupos se generan en el orden de aparición de la clave en la secuencia de entrada. A continuación puede ver un ejemplo que lo explica mejor:

- La clave que define los grupos del componente de origen es el elemento Title. Esta clave sirve para agrupar las personas de la compañía.
- El nombre del grupo se coloca en el elemento Department/Name del componente de destino, mientras que el nombre y el apellido (concatenados) de las personas se colocan en el elemento secundario Person/First.



Observe que group-by utiliza el **contenido** del nodo/elemento como clave de agrupación. El contenido del campo Title se utiliza para agrupar las personas y se asigna al elemento Department/Name del destino.

Además, en el ejemplo anterior existe un **filtro implícito** entre las filas del documento de origen y del documento de destino. En el documento de destino los elementos Department solamente tienen los elementos Person que **coinciden con la clave** de agrupación porque el componente group-by crea la jerarquía necesaria instantáneamente.

Al hacer clic en el panel *Resultados* puede verse el resultado del proceso de agrupación.

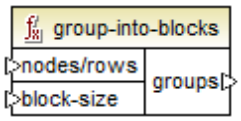
```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <OrgChart xsi:schemaLocation="http://www.xmlspy.com/schemas/orgchart C:/DOCU
3    <Office>
4      <Department>
5        <Name>Office Manager</Name>
6        <Person>
7          <First>Vernon Callaby</First>
8          <First>Steve Meier</First>
9        </Person>
10     </Department>
11     <Department>
12       <Name>Accounts Receivable</Name>
13       <Person>
14         <First>Frank Further</First>
15         <First>Theo Bone</First>
16       </Person>
17     </Department>
```


7.5.8.7 group-ending-with

Agrupa la secuencia del parámetro de entrada **nodes/rows** en grupos, terminando con un grupo nuevo si el parámetro **bool** es `true`.

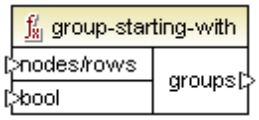
7.5.8.8 group-into-blocks

Agrupa la secuencia de entrada **nodes/rows** en bloques del mismo tamaño. El tamaño de los bloques viene dado por el parámetro **block-size**.

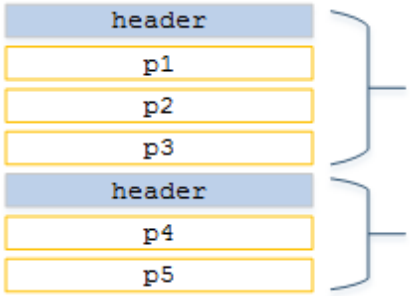


7.5.8.9 group-starting-with

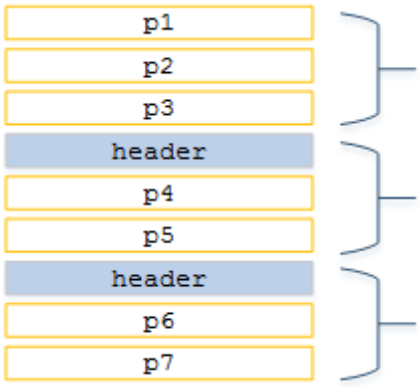
Agrupa la secuencia del parámetro de entrada **nodes/rows** en grupos, empezando con un grupo nuevo si el parámetro **bool** es **true**.



El ejemplo que aparece aquí muestra una secuencia de nodos donde **bool** devuelve `true` cuando se encuentra el nodo "header". Al aplicar la función **group-starting-with** a esta secuencia de nodos se obtienen dos grupos como resultado.

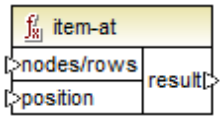


Observe que el primer nodo de la secuencia empieza un grupo nuevo independientemente del valor que tenga **bool**. Es decir, una secuencia como la que aparece a continuación daría lugar a tres grupos.



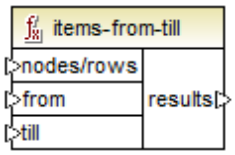
7.5.8.10 item-at

Devuelve los nodos de la secuencia **nodes/rows** situados en la posición dada por el parámetro **position**. El primer elemento está en la posición **1**.



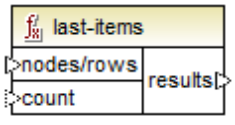
7.5.8.11 items-from-till

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**.



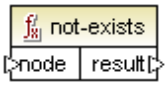
7.5.8.12 last-items

Devuelve los **X** últimos nodos de la secuencia **nodes/rows**, siendo X el número dado por el parámetro **count**. El primer elemento está en la posición **1**.



7.5.8.13 not-exists

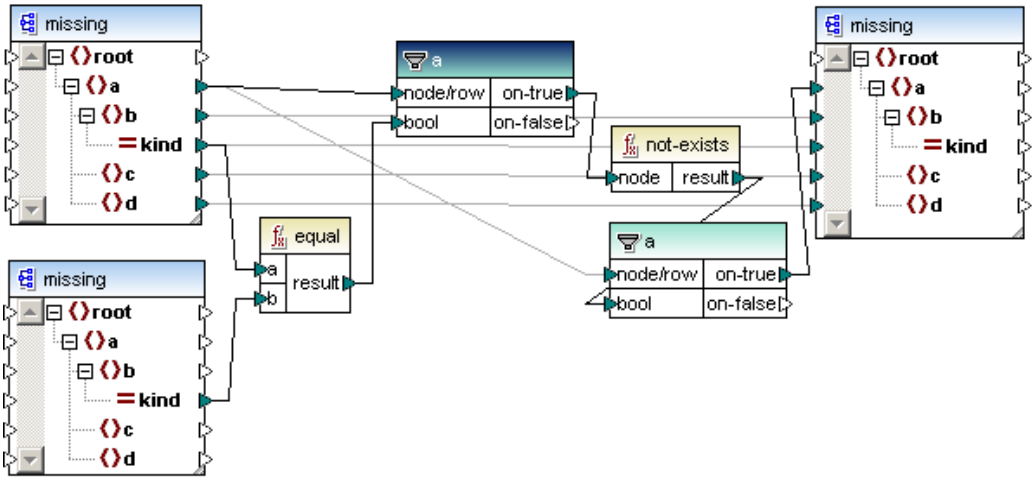
Si el nodo existe, la función devuelve **false**. De lo contrario, devuelve **true**.



En el ejemplo que aparece a continuación se usa la función not-exists para asignar nodos que no existen en uno de los dos archivos de origen.

¿Cómo funciona la asignación del ejemplo?

- Compara los nodos de los dos archivos XML de origen.
- Filtra los nodos del primer archivo XML de origen que no existen en el segundo archivo XML de origen.
- Asigna solamente los nodos que faltan y su contenido al archivo de destino.



Los dos archivos de instancia XML aparecen a continuación. Estas son las diferencias que hay entre los dos archivos:

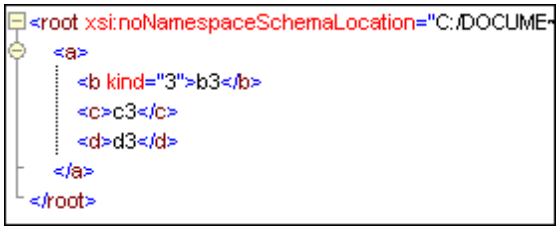
- **a.xml** (izda) contiene el nodo <b **kind="3"**>, que no está en el archivo b.xml.
- **b.xml** (dcha) contiene el nodo <b **kind="4">**, que no está en el archivo a.xml.

```
<root xmlns:xsi="http://www.w3.
    <a>
        <b kind="1">b1</b>
        <c>c1</c>
        <d>d1</d>
    </a>
    <a>
        <b kind="2">b2</b>
        <c>c2</c>
        <d>d2</d>
    </a>
    <a>
        <b kind="3">b3</b>
        <c>c3</c>
        <d>d3</d>
    </a>
</root>
```

```
<root xmlns:xsi="http://www.w3.
    <a>
        <b kind="1">foo</b>
        <c>foo</c>
        <d>foo</d>
    </a>
    <a>
        <b kind="2">foo</b>
        <c>foo</c>
        <d>foo</d>
    </a>
    <a>
        <b kind="4">foo</b>
        <c>foo</c>
        <d>foo</d>
    </a>
</root>
```

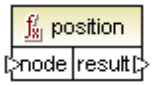
- La función equal compara el tipo de atributo en ambos archivos XML y pasa el resultado al filtro.
- La función not-exists se coloca después del filtro inicial para seleccionar los nodos que faltan en cada uno de los archivos de origen.
- Se usa otro filtro más para pasar solamente el nodo que falta y los demás datos del archivo a.xml al destino.

En el resultado de la asignación el nodo que falta en el archivo b.xml (<b kind="3">) se pasa al componente de destino.



7.5.8.14 position

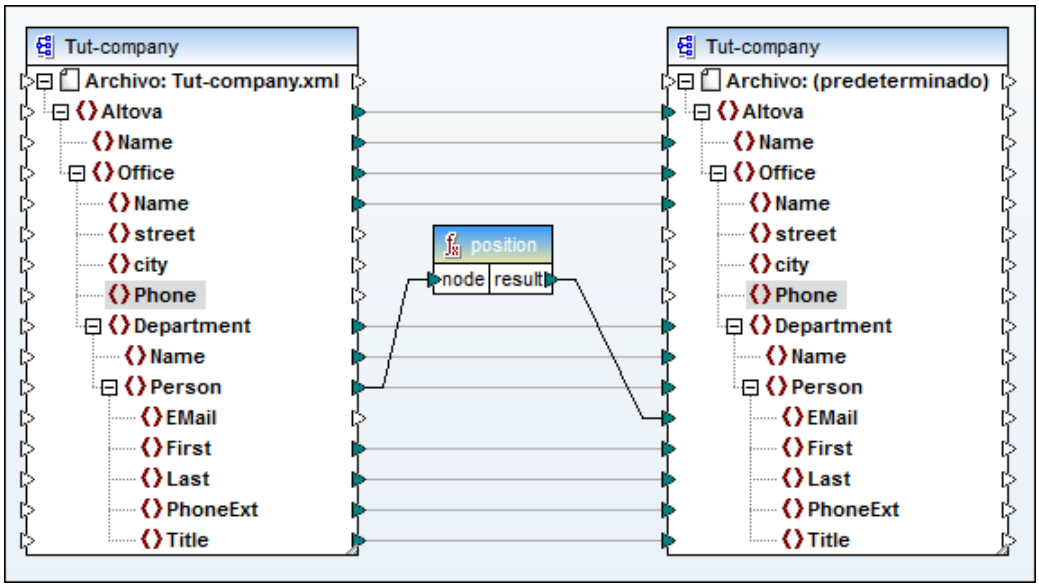
Devuelve la posición de un nodo dentro de la secuencia de la que forma parte.



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.

El elemento de contexto viene definido por el elemento que está conectado al parámetro "node" de la función position (Person).

En la asignación que aparece a continuación se añade un número de posición a cada elemento Person de cada elemento Department.

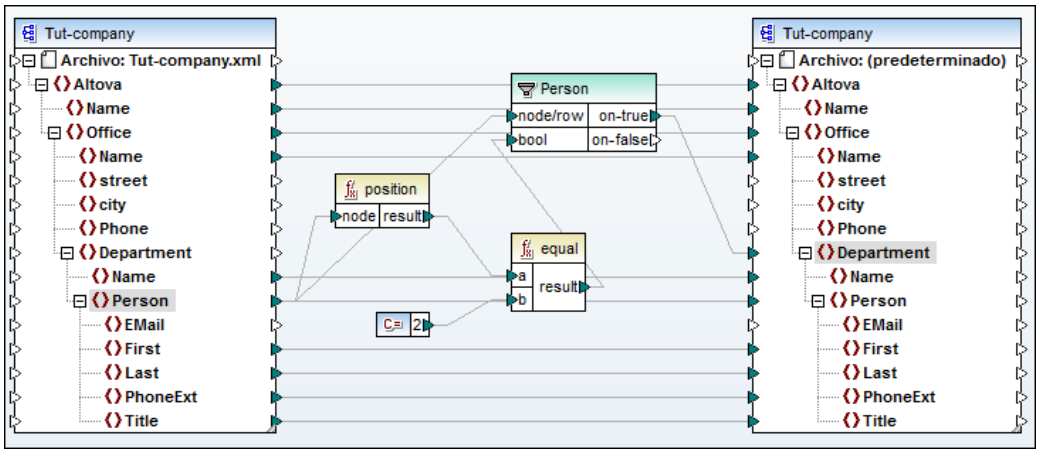


El número de posición se restaura para cada elemento Department del elemento Office.

```
<Office>
  <Name>Microtech, Inc.</Name>
  <Department>
    <Name>Admin</Name>
    <Person>
      <EMail>1</EMail>
      <First>Albert</First>
      <Last>Aldrich</Last>
      <PhoneExt>582</PhoneExt>
      <Title>Manager</Title>
    </Person>
    <Person>
      <EMail>2</EMail>
      <First>Bert</First>
      <Last>Bander</Last>
      <PhoneExt>471</PhoneExt>
      <Title>Accounts Receivable</Title>
    </Person>
```

Si usa la función position junto con un filtro podrá asignar solamente determinados nodos que tengan cierta posición en el componente de origen.

El parámetro de filtro "node/row" y el parámetro "node" de la función position deben estar conectados al mismo elemento del componente de origen para poder filtrar la posición concreta de la secuencia.



El resultado que genera esta asignación es:

- El segundo elemento Person de cada elemento Department de cada elemento Office

```
<Office>
  <Name>Microtech, Inc.</Name>
  <Department>
    <Name>Admin</Name>
    <Person>
      <EMail>b.bander@microtech.com</EMail>
      <First>Bert</First>
      <Last>Bander</Last>
      <Title>Accounts Receivable</Title>
    </Person>
  </Department>
  <Department>
    <Name>Sales and Marketing</Name>
    <Person>
      <EMail>e.ellas@microtech.com</EMail>
      <First>Eve</First>
      <Last>Ellas</Last>
      <Title>Art Director</Title>
    </Person>
  </Department>
  <Department>
    <Name>Manufacturing</Name>
    <Person>
      <EMail>g.gundall@microtech.com</EMail>
```

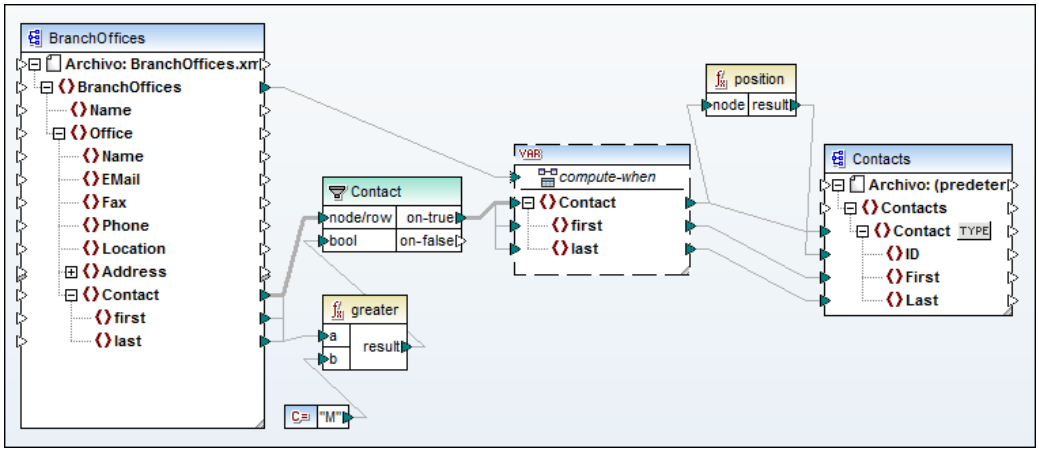
Buscar la posición de los elementos en una secuencia filtrada:

Como el componente de filtrado no es una función de secuencia, no se puede usar directamente con la función position para buscar la posición de los elementos filtrados. En este caso debe utilizarse el componente de variable.

Los resultados de los componentes de variable siempre son secuencias (es decir, listas delimitadas de valores), que se pueden usar para crear secuencias.

- El componente de variable sirve para recopilar los contactos filtrados cuyo apellido empiece con una letra superior a "M".

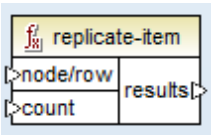
- Después se pasan los contactos (desde la variable) al componente de destino.
- Después la función posicion numera estos contactos de forma secuencial.



7.5.8.15 *replicate-item*

Repeats every item in the input sequence the number of times specified in the `count` argument. If you connect a single item to the `node/row` argument, the function returns N items, where N is the value of the `count` argument. If you connect a sequence of items to the `node/row` argument, the function repeats each individual item in the sequence `count` times, processing one item at a time. For example, if `count` is 2, then the sequence `(1,2,3)` produces `(1,1,2,2,3,3)`.

Repite cada elemento de la secuencia de entrada tantas veces como se indique en el argumento **`count`**. Si conecta un solo elemento al argumento `node/row`, la función devuelve N elementos, siendo N el valor del argumento `count`. Si conecta una secuencia de elementos al argumento `node/row`, la función repite cada uno de los elementos de la secuencia tantas veces como indique `count` y los procesa uno por uno. Por ejemplo, si `count` es 2, entonces la secuencia `(1,2,3)` produce `(1,1,2,2,3,3)`.

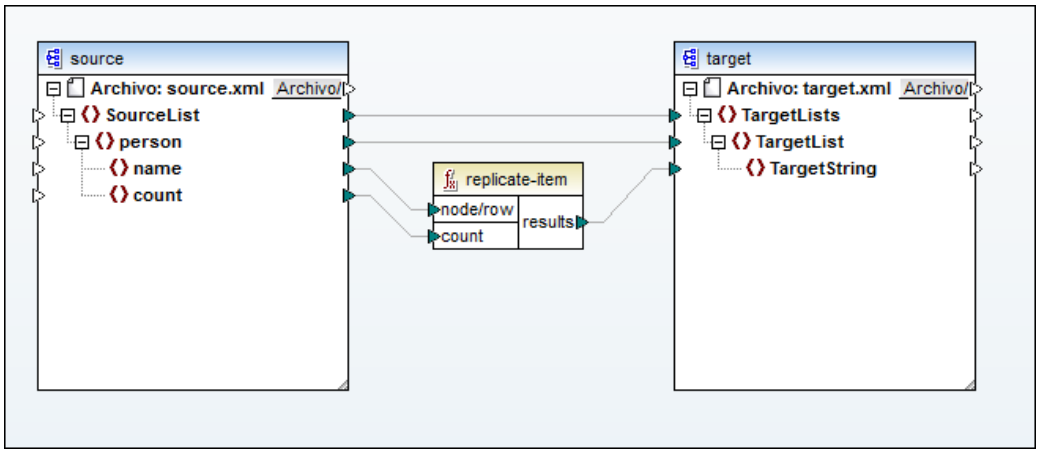


Además, recuerde que puede dar un valor `count` distinto para cada elemento. Por ejemplo, imagine que tiene un archivo XML de origen con esta estructura:

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

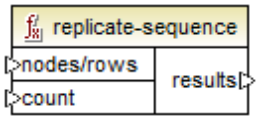
```
<?xml version="1.0" encoding="UTF-8"?>
<TargetLists xsi:noNamespaceSchemaLocation="target.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <TargetList>
      <TargetString>Michelle</TargetString>
      <TargetString>Michelle</TargetString>
   </TargetList>
   <TargetList>
      <TargetString>Ted</TargetString>
      <TargetString>Ted</TargetString>
      <TargetString>Ted</TargetString>
      <TargetString>Ted</TargetString>
   </TargetList>
   <TargetList>
      <TargetString>Ann</TargetString>
      <TargetString>Ann</TargetString>
      <TargetString>Ann</TargetString>
   </TargetList>
```



```
</TargetLists>
```

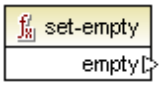
7.5.8.16 replicate-sequence

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)`.



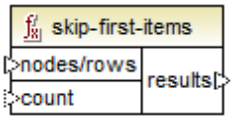
7.5.8.17 set-empty

Devuelve una secuencia vacía.



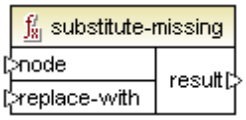
7.5.8.18 skip-first-items

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.



7.5.8.19 substitute-missing

Esta función es una mezcla de la función **exists** y la condición **if-else**. Con ella puede asignar el contenido del campo actual si el nodo existe en el archivo XML de origen. De lo contrario utiliza el elemento asignado al parámetro **replace-with**.



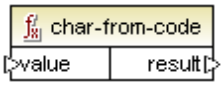
7.5.9 core | string functions (cadena)

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.

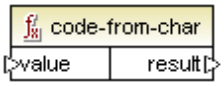
7.5.9.1 char-from-code

El resultado es el carácter que represente el valor Unicode decimal del parámetro **value**.



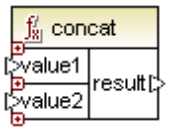
7.5.9.2 code-from-char

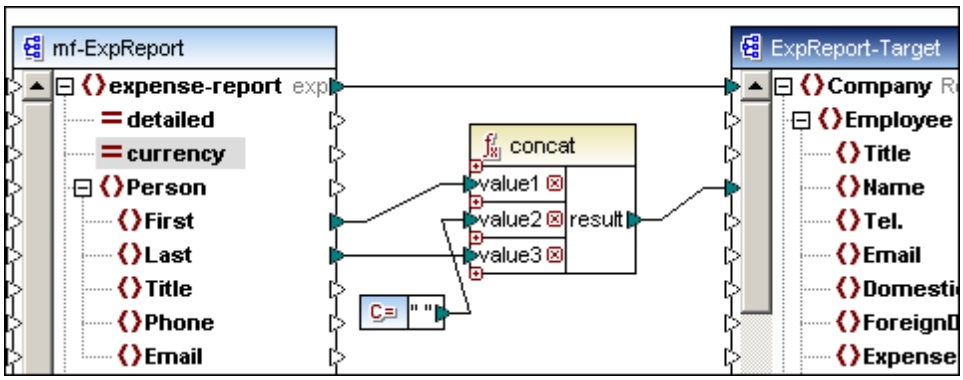
El resultado es el valor Unicode decimal del primer carácter del parámetro **value**.



7.5.9.3 concat

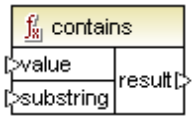
Concatena (anexa) dos o más valores dando lugar a una sola cadena. Todos los valores de entrada se convierten automáticamente en valores de tipo `string`.





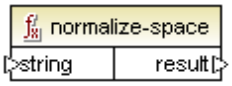
7.5.9.4 contains

El resultado es **true** si los datos asignados al parámetro de entrada **value** contienen la cadena asignada al parámetro de entrada **substring**.



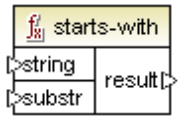
7.5.9.5 normalize-space

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).



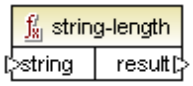
7.5.9.6 starts-with

El resultado es **`true`** si la cadena de entrada **`string`** empieza por la cadena de **`substr`**. De lo contrario, el resultado es **`false`**.



7.5.9.7 string-length

El resultado es el número de caracteres asignados al parámetro de entrada **`string`**.



7.5.9.8 substring

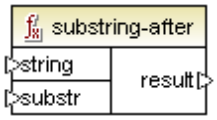
El resultado es la subcadena (fragmento de cadena) del parámetro de entrada **`string`**, siendo **`start`** la posición del carácter de inicio y **`length`** la longitud de la subcadena.

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. Los índices cuentan desde 1.

P. ej. la función **`substring("56789",2,3)`** da como resultado 678.

7.5.9.9 substring-after

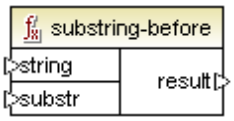
El resultado es el fragmento de la cadena **string** a partir de la primera aparición del parámetro **substr**. Si la cadena del parámetro **substr** no aparece en el parámetro **string**, el resultado de la función es una cadena vacía.



P. ej. la función **substring-after("2009/01/04","/")** da como resultado la subcadena `01/04`.

7.5.9.10 substring-before

El resultado es el fragmento de la cadena del parámetro **string** hasta la primera aparición de los caracteres del parámetro **substr**. Si los caracteres de **substr** no aparecen en **string**, el resultado es una cadena vacía.



P. ej. la función **substring-before ("2009/01/04","/")** da como resultado la subcadena `2009`.

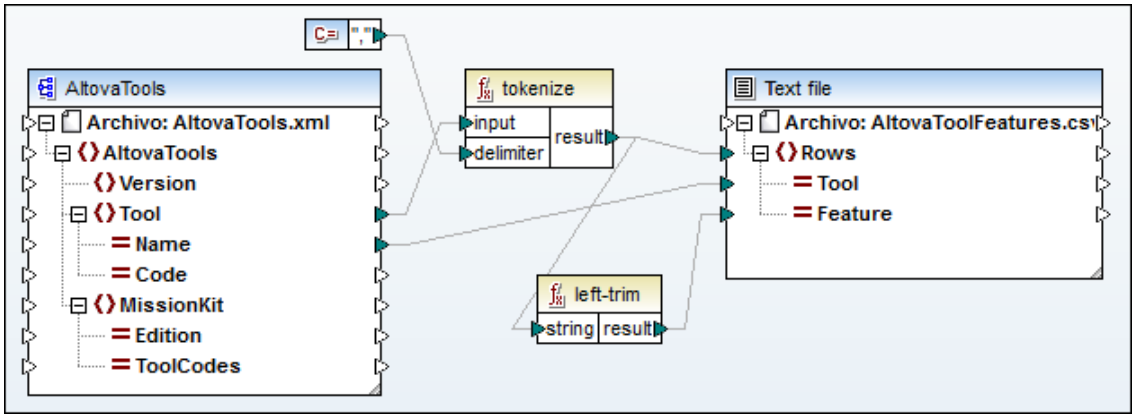
7.5.9.11 tokenize

El resultado es la cadena **input** dividida en una secuencia formada por caracteres delimitados por el parámetro **delimiter**. El resultado se puede seguir procesando.

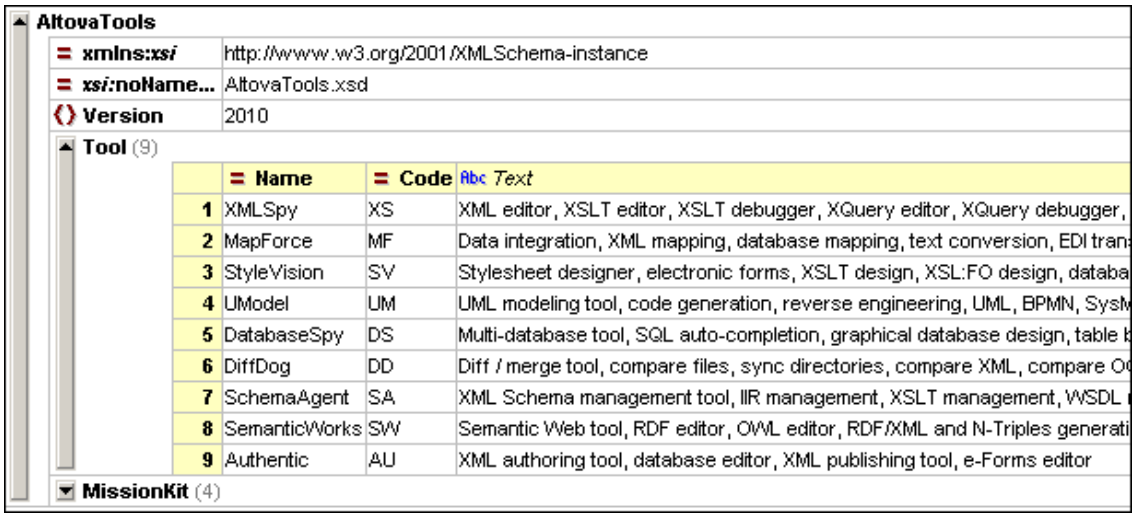
P. ej. la cadena **input** es `A,B,C` y el delimitador es ",". El resultado será A B C.

Ejemplo

En el diseño de asignación de datos **tokenizeString1.mfd** de la carpeta ...\MapForceExamples puede ver un ejemplo de uso de la función tokenize.



A continuación puede ver el archivo XML de origen. El elemento **Tool** tiene dos atributos: Name y Code. Y los datos del elemento **Tool** es texto delimitado por comas.



AltovaTools

xmlns:xsi	http://www.w3.org/2001/XMLSchema-instance
xsi:noName...	AltovaTools.xsd
Version	2010

Tool (9)

	Name	Code	Text
1	XMLSpy	XS	XML editor, XSLT editor, XSLT debugger, XQuery editor, XQuery debugger,
2	MapForce	MF	Data integration, XML mapping, database mapping, text conversion, EDI tran
3	StyleVision	SV	Stylesheet designer, electronic forms, XSLT design, XSL:FO design, databa
4	UModel	UM	UML modeling tool, code generation, reverse engineering, UML, BPMN, Sysl
5	DatabaseSpy	DS	Multi-database tool, SQL auto-completion, graphical database design, table b
6	DiffDog	DD	Diff / merge tool, compare files, sync directories, compare XML, compare O
7	SchemaAgent	SA	XML Schema management tool, IIR management, XSLT management, WSDL
8	SemanticWorks	SW	Semantic Web tool, RDF editor, OWL editor, RDF/XML and N-Triples generati
9	Authentic	AU	XML authoring tool, database editor, XML publishing tool, e-Forms editor

MissionKit (4)

Características de esta asignación de datos

- La función tokenize recibe datos del elemento **Tool** y utiliza el delimitador "**,**" para dividir los datos en secciones (p. ej. la primera sección es "XML editor").
- Como el parámetro result está asignado al elemento **Rows** del componente de destino, se genera una **fila** por cada sección.
- El parámetro result también está asignado a la función **left-trim**, que elimina los espacios en blanco iniciales de cada sección.
- El resultado del parámetro left-trim se asigna al elemento **Feature** del componente de destino.
- El archivo de salida del componente de destino se definió como archivo CSV (AltovaToolFeatures.csv) con el delimitador de campos punto y coma (haga doble clic en el componente para ver su configuración).

Resultado de la asignación de datos

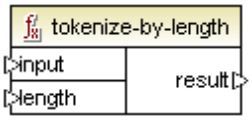
- Por cada elemento Tool del archivo de origen se asigna Name al elemento Tool del componente de destino.
- Cada sección del contenido dividido de Tool se anexa al elemento Feature.
- Por ejemplo, la primera herramienta (XMLSpy) obtiene la primera sección de Feature (XML editor).

- Esto se repite para cada sección del elemento Tool actual y después para todos los elementos Tool.
- A continuación puede ver el resultado en el panel *Resultados*.

```
1   Tool;Feature
2   XMLSpy;XML editor
3   XMLSpy;XSLT editor
4   XMLSpy;XSLT debugger
5   XMLSpy;XQuery editor
6   XMLSpy;XQuery debugger
7   XMLSpy;XML Schema / DTD editor
8   XMLSpy;WSDL editor
9   XMLSpy;SOAP debugger
10  MapForce;Data integration
11  MapForce;XML mapping
12  MapForce;database mapping
```

7.5.9.12 tokenize-by-length

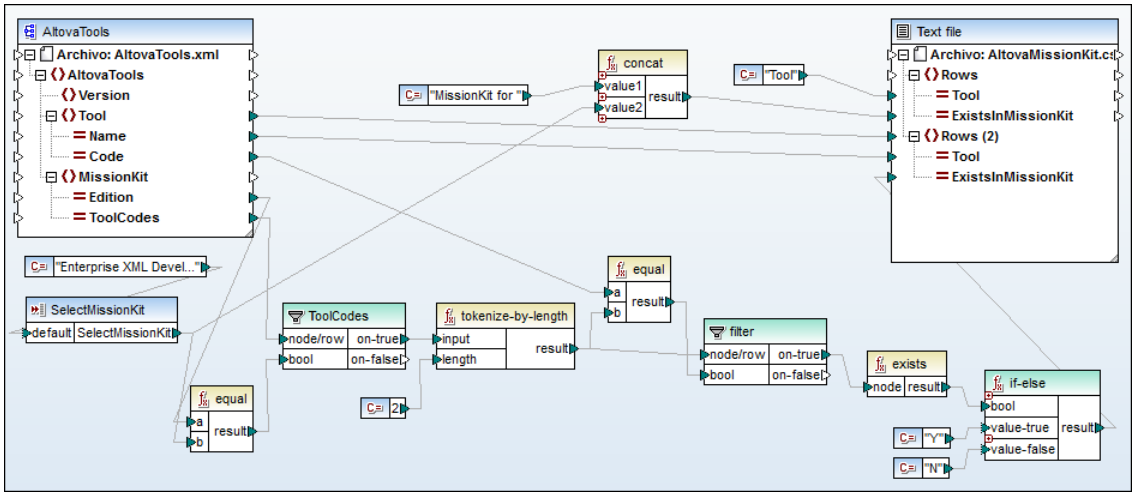
El resultado es la cadena **`input`** dividida en una secuencia formada por tantos caracteres como defina el parámetro **`length`**. El resultado se puede seguir procesando.



P. ej. la cadena **`input`** es ABCDEF y el parámetro **`length`** es 2 . El resultado es AB CD EF.

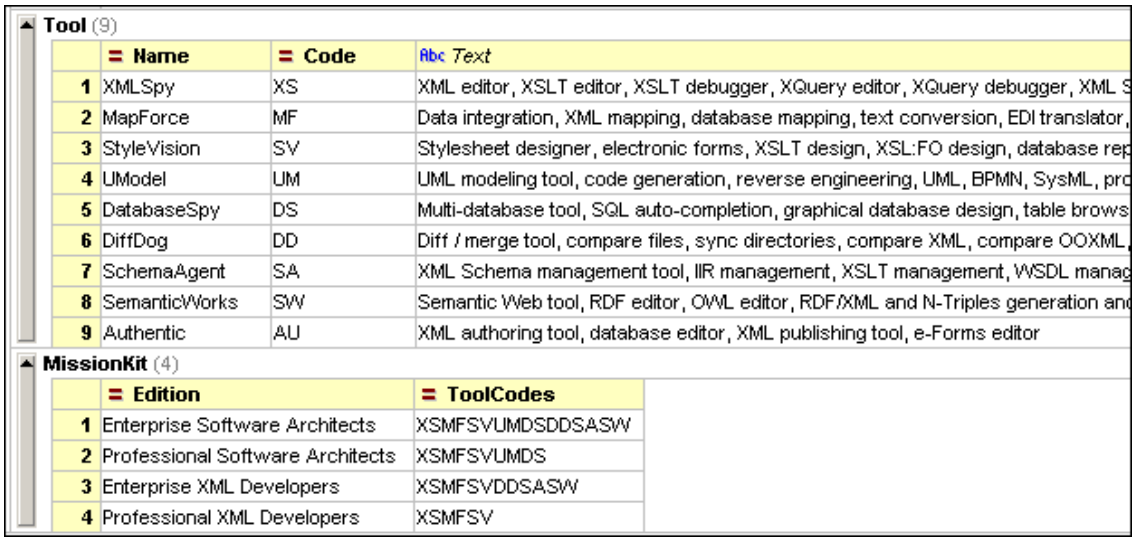
Ejemplo

En el diseño de asignación de datos **tokenizeString2.mfd** de la carpeta ...\MapForceExamples puede ver un ejemplo de uso de la función **tokenize-by-length**.



El archivo XML de origen se puede ver más abajo y es el mismo que el del ejemplo anterior.

El elemento **MissionKit** también tiene dos atributos (**Edition** y **ToolCodes**) pero no tiene contenido.



Tool (9)

	= Name	= Code	Abc Text
1	XMLSpy	XS	XML editor, XSLT editor, XSLT debugger, XQuery editor, XQuery debugger, XML S
2	MapForce	MF	Data integration, XML mapping, database mapping, text conversion, EDI translator,
3	StyleVision	SV	Stylesheet designer, electronic forms, XSLT design, XSL:FO design, database rep
4	UModel	UM	UML modeling tool, code generation, reverse engineering, UML, BPMN, SysML, pro
5	DatabaseSpy	DS	Multi-database tool, SQL auto-completion, graphical database design, table brows
6	DiffDog	DD	Diff / merge tool, compare files, sync directories, compare XML, compare OOXML,
7	SchemaAgent	SA	XML Schema management tool, IIR management, XSLT management, WSDL manag
8	SemanticWorks	SW	Semantic Web tool, RDF editor, OWL editor, RDF/XML and N-Triples generation and
9	Authentic	AU	XML authoring tool, database editor, XML publishing tool, e-Forms editor

MissionKit (4)

	= Edition	= ToolCodes
1	Enterprise Software Architects	XSMFSVUMDSDDSASW
2	Professional Software Architects	XSMFSVUMDS
3	Enterprise XML Developers	XSMFSVDDSASW
4	Professional XML Developers	XSMFSV

Objetivo de la asignación

Generar una lista que muestre qué herramientas de Altova forman parte de cada edición de MissionKit.

Características de esta asignación de datos

- El componente de entrada **SelectMissionKit** recibe su entrada predeterminada de una constante ("Enterprise XML Developers").
- La función **equal** compara el valor de entrada con el valor "**Edition**" y pasa el resultado al parámetro **bool** del filtro ToolCodes.
- La entrada **node/row** del filtro ToolCodes viene dada por el elemento **ToolCodes** del archivo de origen. El valor para la edición Enterprise XML Developers es: XSMFSVDDSASW.
- El valor XSMFSVDDSASW se envía al parámetro **on-true** y después al parámetro **input** de la función **tokenize-by-length**.

¿Qué hace la función tokenize-by-length?

- El valor de entrada ToolCodes es XSMFSVDDSASW y se divide en varias secciones de dos caracteres cada una (definidas por el parámetro **length** que es 2, lo cual da lugar a 6 secciones).
- Cada sección (situada en el parámetro b) de la función equal se compara con el valor **Code** de dos caracteres del archivo de origen (del cual existen 9 entradas en total).
- El resultado de la comparación (true/false) se envía al parámetro **bool** del filtro.
- **Todas** las secciones de la función tokenize-by-length se envían al parámetro **node/row** del filtro.
- La función **exists** busca los nodos que existen o que faltan que recibió del parámetro **on-true** del componente de filtrado.

  Los nodos que existen son aquellos en los que la sección **ToolCodes** y el valor Code **coinciden**.

  Los nodos que faltan son aquellos donde **ninguna sección ToolCodes coincide** con ningún valor Code.

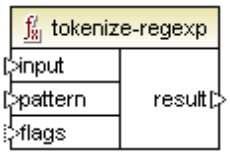
- Los resultados bool de la función **exists** se pasan a la función **if-else** que pasa una Y al destino si el nodo existe y una N si no existe.

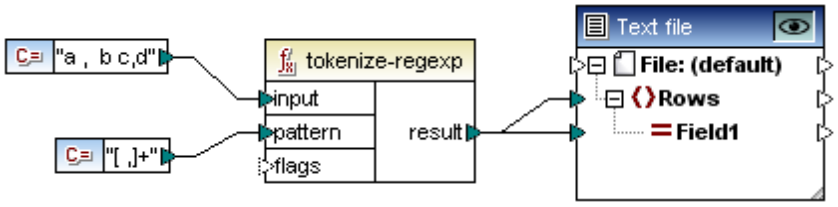
Resultado de la asignación

```
1   Tool;MissionKit for Enterprise XML Developers
2   XMLSpy;Y
3   MapForce;Y
4   StyleVision;Y
5   UModel;N
6   DatabaseSpy;N
7   DiffDog;Y
8   SchemaAgent;Y
9   SemanticWorks;Y
10  Authentic;N
11
```

7.5.9.13 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. También puede usar marcas opcionales.





En el ejemplo anterior la cadena **input** es una serie de caracteres separados por espacios y/o comas (es decir, `a , b c,d`).

La expresión regular **pattern** define una clase de caracteres **[**"espacio""coma"**]** 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.

La cadena resultante es:

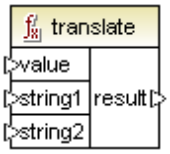
1	a
2	b
3	c
4	d
5	

Recuerde que la sintaxis de las expresiones regulares varía ligeramente de un lenguaje a otro. La función **tokenize-regexp** en C++ solamente está disponible en Visual Studio 2008 SP1 (o superior).

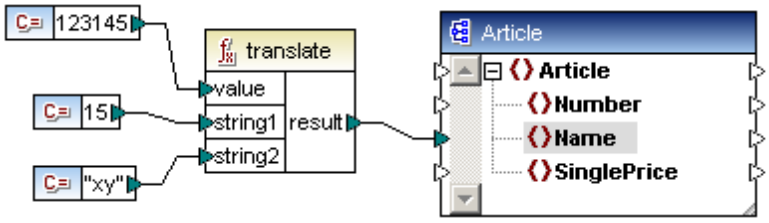
Consulte el apartado Expresiones regulares para más información.

7.5.9.14 translate

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.



Por ejemplo:
-la cadena de entrada **value** es 123145
-la cadena de búsqueda **string1** es 15
-la cadena de reemplazo **string2** es xy

Como resultado:
-cada carácter **1** se reemplaza con **x** en la cadena de entrada **value**
-cada carácter **5** se reemplaza con **y** en la cadena de entrada **value**

La cadena resultante es **x23x4y**

Si el parámetro **string2** está vacío, el carácter de **string1** se elimina.

Por ejemplo:
- la cadena de entrada **value** es aabaacbca
- la cadena de búsqueda **string1** es "**a**"
- la cadena de reemplazo **string2** es "" (una cadena vacía)

La cadena resultante es **bcbc**

Otro ejemplo:
-la cadena de entrada **value** es aabaacbca
-la cadena de búsqueda **string1** es "**ac**"
-la cadena de reemplazo **string2** es "**ca**"

La cadena resultante es **ccbccabac**

7.5.10 xpath2 | accessors (descriptores de acceso)

La biblioteca **xpath2** contiene funciones para trabajar con los lenguajes XSLT2 y XQuery.

7.5.10.1 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 resultado es de tipo `xs:string`. MapForce devuelve un error si no se suministra un nodo de entrada.

7.5.10.2 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 de texto, tiene el formato `prefijo:nombrelocal` (si el nodo tiene un prefijo) o el formato `nombrelocal` (si el nodo no tiene prefijo). Para obtener el URI de espacio de nombres de un nodo, utilice la función **namespace-URI-from-QName** (de la biblioteca de funciones **xpath2 | qname-related**).

7.5.10.3 string

La función **string** funciona igual que 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.)

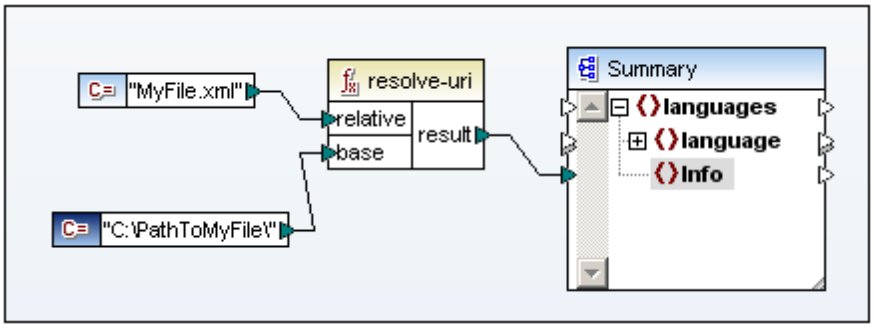
7.5.11 xpath2 | anyURI functions

La biblioteca **xpath2** contiene funciones para trabajar con los lenguajes XSLT2 y XQuery.

7.5.11.1 resolve-uri

La función **resolve-uri** toma un URI como primer argumento (tipo de datos `xs:string`) y lo resuelve a partir del URI del segundo argumento (tipo de datos `xs:string`).

El resultado (tipo de datos `xs:string`) es un URI combinado. De este modo se puede convertir un URI relativo (primer argumento) en un URI absoluto, resolviéndolo a partir de un URI base.



En el ejemplo de esta imagen el primer argumento aporta el URI relativo y el segundo aporta el URI base. El URI resuelto será una concatenación del URI base y el URI relativo. Es decir `C:\PathtoMyFile\MyFile.xml`.

Nota: ambos argumentos son de tipo de datos `xs:string` y para el proceso de combinación ambos datos de entrada se tratan como cadenas de texto. Por tanto, no se puede comprobar si los recursos identificados por estos URI existen o no. MapForce devuelve un error si falta el segundo argumento.

7.5.12 xpath2 | boolean functions (booleanas)

La biblioteca **xpath2** contiene funciones para trabajar con los lenguajes XSLT2 y XQuery.

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.

7.5.12.1 false

Devuelve el valor booleano `false`.

7.5.12.2 true

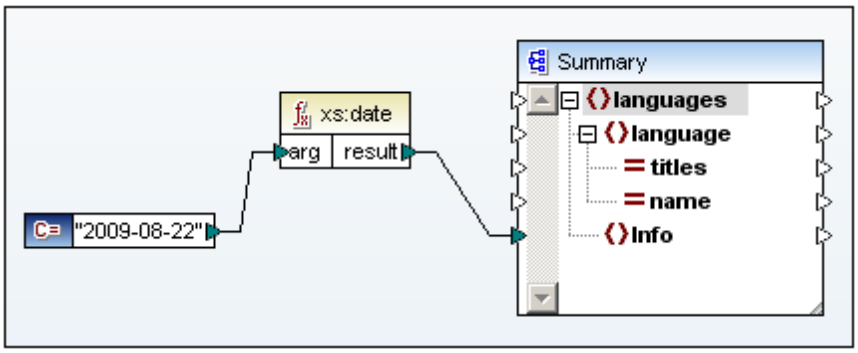
Devuelve el valor booleano `true`.

7.5.13 xpath2 | constructors (constructores)

La biblioteca **xpath2** contiene funciones para trabajar con los lenguajes XSLT2 y XQuery.

Las funciones constructores de la biblioteca de funciones **xpath2** construyen ciertos tipos de datos a partir del texto de entrada. 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:date`, utilice la función constructor `xs:date`. El texto de entrada debe tener el formato léxico del tipo de datos `xs:date`, es decir: `AAAA-MM-DD` (*imagen siguiente*).



En la imagen anterior, observe que se utilizó la constante `2009-08-22` como argumento de entrada de la función. La entrada también podría obtenerse de un nodo de un documento de origen.

La función `xs:date` devuelve el texto de entrada (`2009-08-22`), que es de tipo de datos `xs:string` (especificado en la constante), como tipo de datos `xs:date` de salida.

Al pasar el puntero del ratón encima del argumento de entrada de la función aparece el tipo de datos esperado para el argumento.

7.5.14 xpath2 | context functions (contexto)

La biblioteca **xpath2** contiene funciones para trabajar con los lenguajes XSLT2 y XQuery.

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.

7.5.14.1 current-date

Devuelve la fecha actual (`xs:date`) del reloj del sistema.

7.5.14.2 current-dateTime

Devuelve la fecha y la hora actuales (`xs:dateTime`) del reloj del sistema.

7.5.14.3 current-time

Devuelve la hora actual (`xs:time`) del reloj del sistema.

7.5.14.4 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.

El motor XSLT 2.0 de Altova solamente es compatible con la intercalación de punto de código Unicode. Las comparaciones (incluidas las funciones **fn:max** y **fn:min**) se basan en esta intercalación.

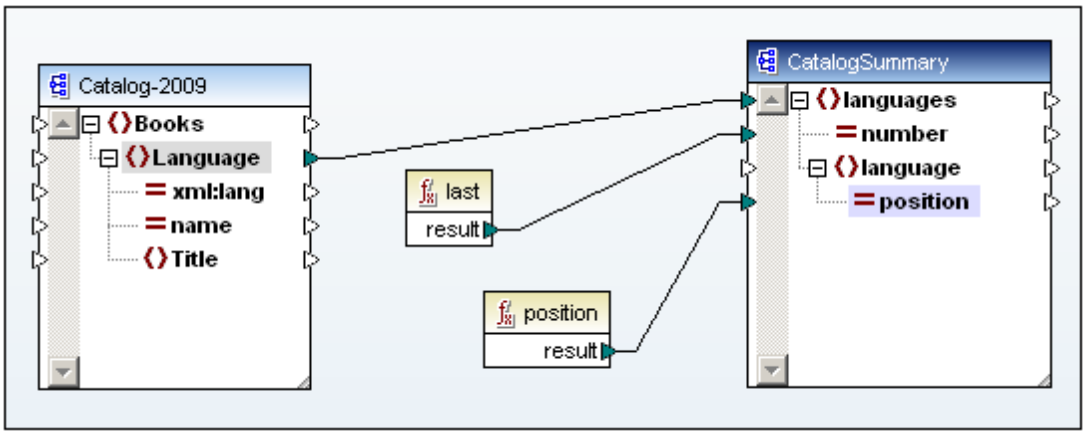
7.5.14.5 implicit-timezone

Devuelve el valor de la propiedad de zona horaria implícita "*implicit timezone*" a partir del contexto de evaluación.

7.5.14.6 last

Las funciones **last** y **position** tampoco tienen argumentos. La función **last** devuelve la posición del último nodo del conjunto de nodos de contexto. La función **position** devuelve la posición del nodo actual del conjunto de nodos que se está procesando.

El conjunto de nodos de contexto de los nodos a los que se dirigen las funciones es el conjunto de nodos al que afectarán las funciones. Por ejemplo, en la imagen siguiente, el conjunto de nodos de los elementos `Language` es el conjunto de nodos de contexto para las funciones **last** y **position**.



En el ejemplo, la función **last** devuelve la posición del último nodo del conjunto de nodos de contexto (el conjunto de nodos de los elementos `Language`) como valor del atributo `number`. Este valor también es el tamaño del conjunto de nodos, puesto que indica el número de nodos que contiene el conjunto.

La función **position** devuelve la posición del nodo `Language` que se está procesando en ese momento. La posición de cada nodo `Language` dentro del conjunto de nodos de los elementos `Language` se escribe en el atributo `language/@position`.

Recomendamos utilizar las funciones **position** y **count** de la biblioteca de funciones **core**.

7.5.15 xpath2 | durations, date, time functions (duración, fecha y hora)

La biblioteca **xpath2** contiene funciones para trabajar con los lenguajes XSLT2 y XQuery.

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.

Funciones Adjust-to-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.

Cuando el primer argumento no contiene datos de uso horario (por ejemplo, la fecha `2009-01` o la hora `14:00:00`), hay tres posibilidades:

- Si está presente el argumento `timezone` (el segundo argumento de la función), el resultado contendrá el uso horario especificado en el segundo argumento.
- Si falta el argumento `timezone` (el segundo argumento de la función), el resultado contendrá el uso horario implícito, es decir, el del sistema.
- Si el argumento `timezone` (el segundo argumento de la función) está vacío, el resultado no contendrá el uso horario.

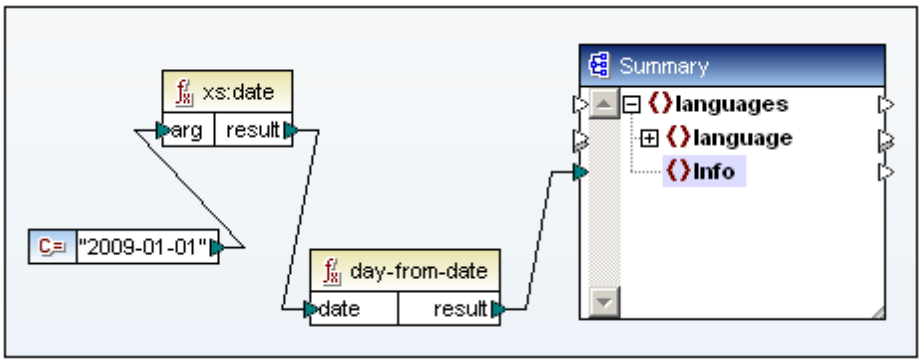
Cuando el primer argumento contiene datos de uso horario (por ejemplo, la fecha `2009-01-01 +01:00` o la hora `14:00:00+01:00`), hay tres posibilidades:

- Si está presente el argumento `timezone` (el segundo argumento de la función), el resultado contendrá el uso horario especificado en el segundo argumento. El uso horario original se sustituye con el uso horario del segundo argumento.
- Si falta el argumento `timezone` (el segundo argumento de la función), el resultado contendrá el uso horario implícito, es decir, el del sistema. El uso horario original se sustituye con el uso horario del sistema.
- Si el argumento `timezone` (el segundo argumento de la función) está vacío, el resultado no contendrá el uso horario.

Funciones From

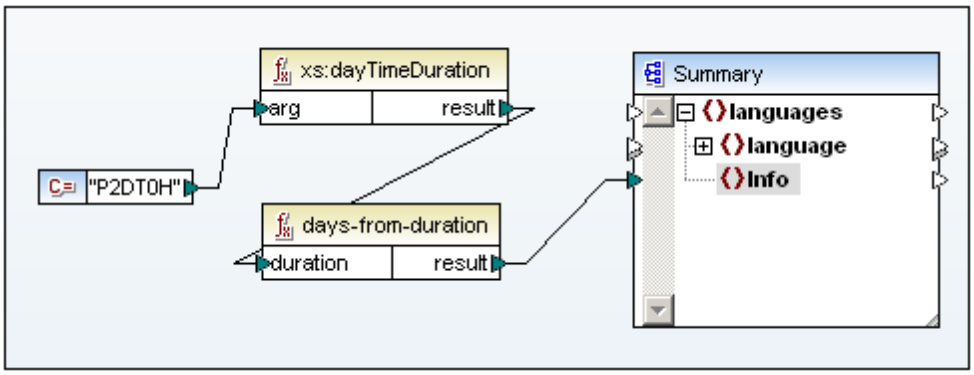
Cada función **From** extrae un componente de los (i) datos de fecha u hora o de los (ii) datos de duración. Los resultados son de tipo de datos `xs:decimal`.

Por ejemplo, si queremos extraer un componente de los datos de fecha u hora, podemos usar la función **day-from-date** (*imagen siguiente*).



El argumento de entrada es una fecha (`2009-01-01`) de tipo `xs:date`. La función **`day-from-date`** extrae el componente día de la fecha (`1`) con tipo de datos `xs:decimal`.

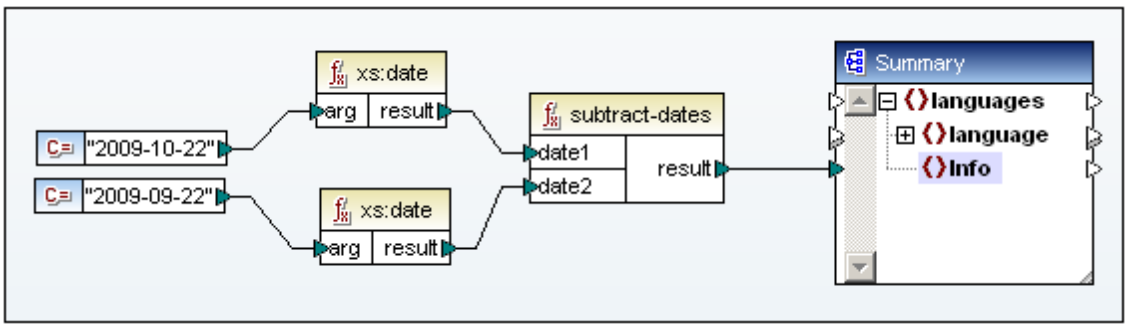
Para extraer los componentes hora de las duraciones es necesario indicar la duración como `xs:yearMonthDuration` (para extraer años y meses) o como `xs:dayTimeDuration` (para extraer días, horas, minutos y segundos). El resultado será de tipo `xs:decimal`. En el ejemplo siguiente, los datos de entrada de la función **`days-from-duration`** es el tipo de datos `dayTimeDuration` de `P2DT0H`. El resultado es el `2`, de tipo de datos `xs:decimal`.



Funciones Subtract

Hay tres funciones de sustracción con las que puede sustraer un valor de hora a otro y obtener un valor de duración como resultado. Se trata de las funciones: **`subtract-dates`**, **`subtract-times`** y **`subtract-dateTimes`**.

En el ejemplo de la imagen se usa la función **`subtract-dates`** para sustraer una fecha a otra (`2009-10-22` menos `2009-09-22`). El resultado es `P30D`, de tipo de datos `dayTimeDuration`.

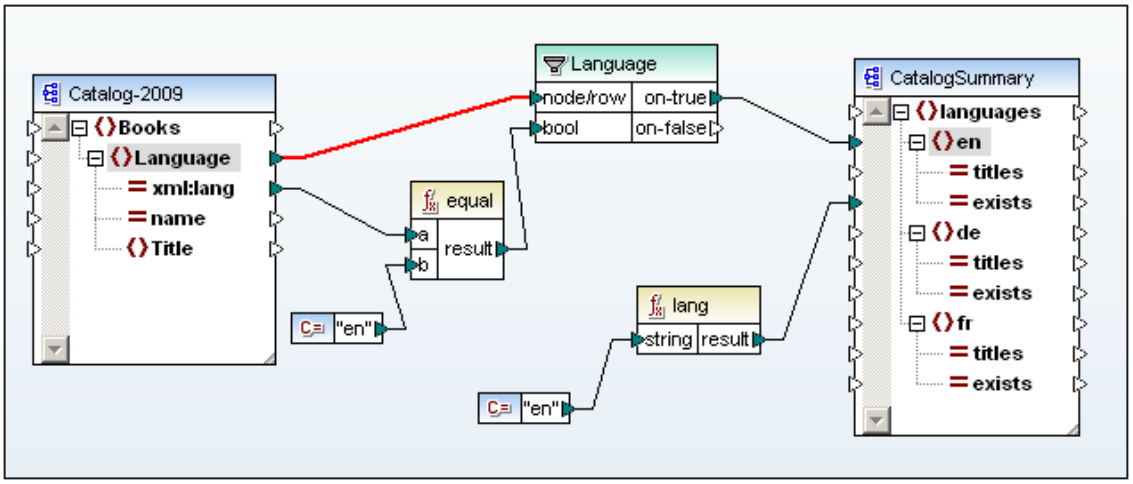


7.5.16 xpath2 | node functions (nodo)

La biblioteca **xpath2 | node** incluye estas funciones de nodo:

lang

Esta función 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.



Por ejemplo, en la imagen anterior:

1. En el esquema de origen el elemento `Language` tiene un atributo `xml:lang`.
2. Los nodos `Language` se filtran para que se procesen solamente los nodos `Language` cuyo atributo `xml:lang` tenga el valor `en` (la prueba de filtro se especifica en la función **`equal`**).
3. El nodo `Language` es el nodo de contexto donde se crea el elemento `en` en el documento de salida.
4. El resultado de la función **`lang`** (`true` o `false`) se envía al atributo `en/@exists` de salida. El argumento de la función viene dado por la constante `en`. La función **`lang`** comprueba si el nodo de contexto en este punto (el elemento `Language`) tiene un atributo `xml:lang` con valor `en` (el argumento de la función). Si lo tiene, la función devuelve `true`. Si no, devuelve `false`.

local-name, name, namespace-uri

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:` (por ejemplo `https://www.altova.com/mapforce`).

Cada una de estas tres funciones tiene dos variantes:

- Sin argumento: la función se aplica al nodo de contexto (*imagen anterior*).
- Con argumento (que debe ser un nodo): la función se aplica al nodo especificado.

El resultado de estas tres funciones (en todas sus variantes) es una cadena de texto.

number

Esta función convierte la cadena de entrada en un número. También convierte valores de entrada booleanos en un número.

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. Los tipos que se pueden convertir en números son los tipos `boolean`, `string` y otros tipos numéricos. Los valores de entrada no numéricos (como cadenas no numéricas) dan como resultado `NaN` (no es un número).

Esta función tiene dos variantes:

- Sin argumento: la función se aplica al nodo de contexto.
- Con argumento (que debe ser un nodo): la función se aplica al nodo especificado.

7.5.17 xpath2 | numeric functions (numéricas)

La biblioteca de funciones **xpath2 | numeric** ofrece estas funciones numéricas:

abs

Esta función toma un valor numérico como entrada y devuelve su valor absoluto como `decimal`. Por ejemplo, si el argumento de entrada es `-2` o `+2`, la función devuelve `2`.

round-half-to-even

Esta función redondea el número dado (primer argumento) al grado de precisión (número de decimales) dado en el segundo argumento, que es opcional. Por ejemplo, si el primer argumento es `2.141567` y el segundo argumento es `3`, el primer argumento (el número) se redondea con tres decimales y el resultado sería `2.141`. Si no se especifica la precisión (si falta el segundo argumento), el número se redondea con cero decimales, es decir, se convierte en un número entero.

La parte "even" del nombre de la función hace referencia a que se redondeo hasta un número par cuando un dígito del número dado está entre dos valores. Por ejemplo `round-half-to-even(3.475, 2)` devolvería el valor `3.48`.

7.5.18 xpath2 | string functions (cadena)

La biblioteca **xpath2 | string** ofrece estas funciones de cadena:

compare

Esta función toma dos cadenas como argumentos y las compara alfabéticamente. Si la cadena `String-1` es alfabéticamente menor que la cadena *String-2* (por ejemplo: `A y B`), el resultado de la función es -1. Si las dos cadenas son iguales (por ejemplo, `A` y `A`), la función devuelve `0`. Si la cadena `String-1` es mayor que `String-2` (por ejemplo, `B y A`), la función devuelve +1.

Esta función tiene una variante que sirve para elegir la intercalación que se debe utilizar para comparar las cadenas. Cuando no se indica la intercalación, se utiliza la intercalación predeterminada (la intercalación de punto de código Unicode).

ends-with

Esta función comprueba si la cadena `String-1` termina con la cadena `String-2`. Si es así, devuelve el valor `true`. Si no, devuelve `false`.

Esta función tiene una variante que sirve para elegir la intercalación que se debe utilizar para comparar las cadenas. Cuando no se indica la intercalación, se utiliza la intercalación predeterminada (la intercalación de punto de código Unicode). Los motores de Altova solamente son compatibles con la intercalación de punto de código Unicode.

escape-uri

Esta función toma un URI como entrada para el primer argumento de cadena y aplica a la cadena las normas de escape de URI RFC 2396. El segundo argumento booleano (`escape-reserved`) debe ser `true()` si los caracteres con significado reservado en los URI se deben escapar (por ejemplo "+" o "/").

Ejemplos:

`escape-uri("My A+B.doc", true())` daría como resultado `My%20A%2B.doc`
`escape-uri("My A+B.doc", false())`daría como resultado `My%20A+B.doc`

lower-case

Esta función toma una cadena como argumento y pone en minúsculas todos los caracteres que estén en mayúsculas.

matches

Esta función comprueba si la cadena de entrada (primer argumento) coincide con una expresión regular (segundo argumento). La sintaxis de las expresiones regulares debe ser la sintaxis definida para la faceta `pattern` de XML Schema. La función devuelve `true` si la cadena coincide con la expresión regular. De lo contrario, el resultado es `false`.

La función toma un argumento opcional llamado `flags`. Este argumento `flags` puede tener cuatro valores (`i`, `m`, `s`, `x`) y se pueden usar varios a la vez (p. ej. `imx`) Si se deja vacío, se usan los valores predeterminados.

El significado de estos argumentos es:

- `i` Utilizar el modo de no distinción entre mayúsculas y minúsculas. El modo predeterminado es el modo de distinción entre mayúsculas y minúsculas.
- `m` Usar el modo multilínea, que entiende que la cadena de entrada tiene varias líneas, cada una de ellas separada por un carácter de línea nueva (`x0a`). Los metacaracteres `^` y `$` indican el comienzo y el final de cada línea. El modo predeterminado es el modo de cadena, en el que la cadena comienza y termina con los metacaracteres `^` y `$`.
- `s` Usar el modo *dot-all*. El modo predeterminado es el modo *not-dot-all*, en el que el metacarácter "." coincide con todos los caracteres excepto el carácter de línea nueva

(`x0a`). En el modo *dot-all*, el punto **"."** también coincide con el carácter de línea nueva.

`x` Omitir espacios en blanco. Los caracteres de espacio en blanco no se ignoran por defecto.

normalize-unicode

Esta función normaliza la cadena de entrada (primer argumento) según las reglas de normalización especificadas (segundo argumento). Son compatibles los formatos de normalización NFC, NFD, NFKC y NFKD.

replace

Esta función toma la cadena del primer argumento como entrada, busca en ella repeticiones de la expresión regular (segundo argumento) y reemplaza los resultados encontrados con la cadena del tercer argumento.

Las reglas de coincidencia son las indicadas para el atributo `matches`. La función también toma el argumento opcional `flags`. Los valores de este argumento son los mismos que los de la función **`matches`** (*ver más arriba*).

starts-with

Esta función comprueba si la cadena `String-1` empieza con la cadena `String-2`. Si es así, la función devuelve `true`. De lo contrario devuelve `false`.

Esta función tiene una variante que sirve para elegir la intercalación que se debe utilizar para comparar las cadenas. Cuando no se indica la intercalación, se utiliza la intercalación predeterminada (la intercalación de punto de código Unicode). Los motores de Altova solamente son compatibles con la intercalación de punto de código Unicode.

substring-after

Esta función devuelve la parte de la cadena `String-1` (primer argumento) que aparece después de la cadena de prueba `String-2` (segundo argumento). La función toma un tercer argumento opcional que especifica la intercalación que se debe usar para la comparación de las cadenas. Cuando no se indica la intercalación, se utiliza la intercalación predeterminada (la intercalación de punto de código Unicode). Los motores de Altova solamente son compatibles con la intercalación de punto de código Unicode.

substring-before

Esta función devuelve la parte de la cadena `String-1` (primer argumento) que aparece antes de la cadena de prueba `String-2` (segundo argumento). La función toma un tercer argumento opcional que especifica la intercalación que se debe usar para la comparación de las cadenas. Cuando no se indica la intercalación, se utiliza la intercalación predeterminada (la intercalación de punto de código Unicode). Los motores de Altova solamente son compatibles con la intercalación de punto de código Unicode.

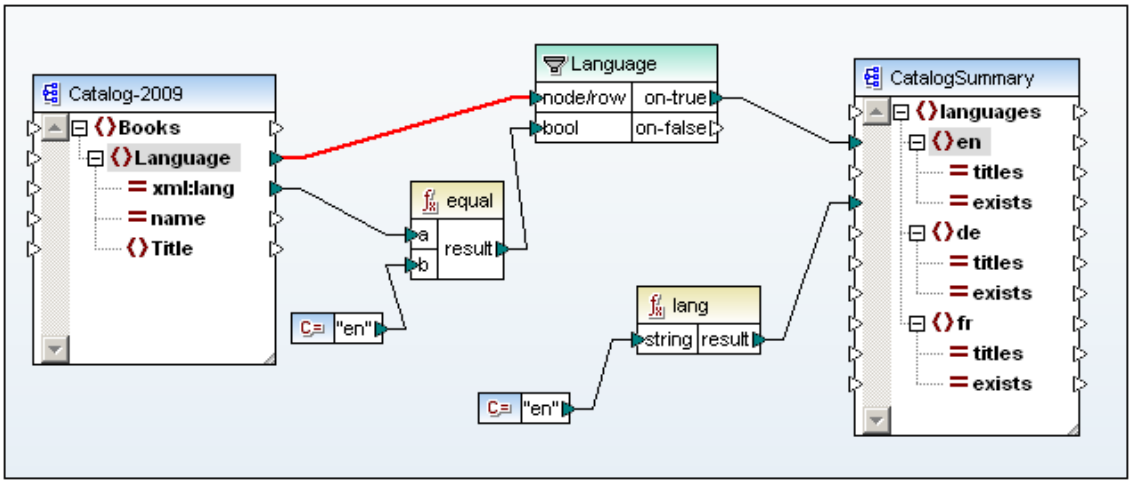
upper-case

Esta función toma una cadena y pone en mayúsculas todos los caracteres que aparezcan en minúsculas.

7.5.19 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. Estas funciones suelen tener:

- un nodo de contexto (por ejemplo, el nodo de contexto para la función **lang** de la imagen siguiente es el elemento `Language` del esquema de origen).
- un argumento de entrada (por ejemplo, el argumento de entrada para la función **lang** de la imagen siguiente es la constante `en`). Las funciones `last` y `position` no toman argumentos.



lang

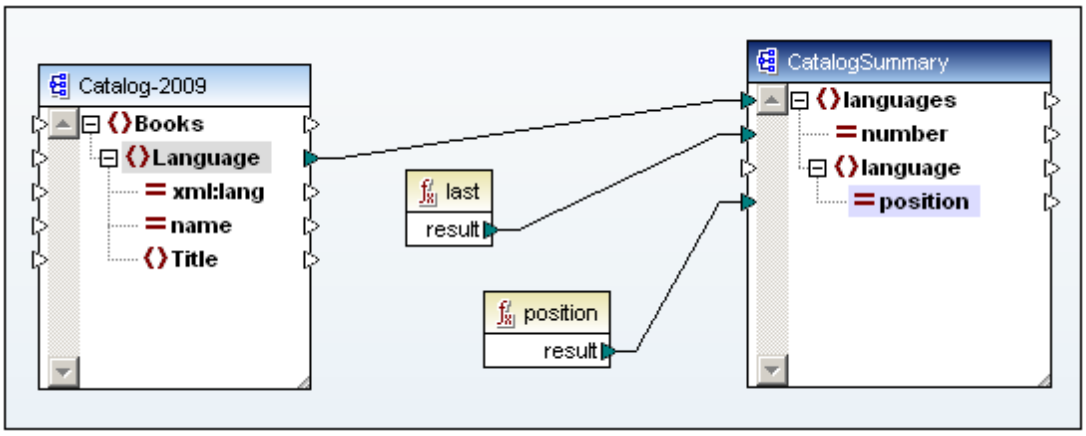
Esta función toma una cadena de entrada que identifica el código de idioma (p. ej. `en`). La función devuelve `true` o `false` dependiendo de si el nodo de contexto tiene o no un atributo `xml:lang` cuyo valor coincide con el argumento de la función. En la imagen anterior, por ejemplo:

1. El elemento `Language` del esquema de origen tiene un atributo `xml:lang`.
2. Los nodos `Language` se filtran para que solamente se procesen aquellos cuyo atributo `xml:lang` tenga el valor `en` (la prueba de filtro se especifica en la función **`equal`**).
3. El nodo `Language` es el nodo de contexto en el punto donde se crea el elemento `en` en el documento de destino.
4. El resultado de la función **`lang`** (`true` o `false`) se envía al atributo `en/@exists` del documento de destino. El argumento de la función viene dado por la constante `en`. La función **`lang`** comprueba si el nodo de contexto en este punto (el elemento `Language`) tiene o no un atributo `xml:lang` con valor `en` (el argumento de la función). Si es así, la función devuelve `true`. Si no, devuelve `false`.

last, position

Las funciones **`last`** y **`position`** tampoco tienen argumentos. La función **`last`** devuelve la posición del último nodo del conjunto de nodos de contexto. La función **`position`** devuelve la posición del nodo actual del conjunto de nodos que se está procesando.

El conjunto de nodos de contexto de los nodos a los que se dirigen las funciones es el conjunto de nodos al que afectarán las funciones. Por ejemplo, en la imagen siguiente, el conjunto de nodos de los elementos `Language` es el conjunto de nodos de contexto para las funciones **`last`** y **`position`**.



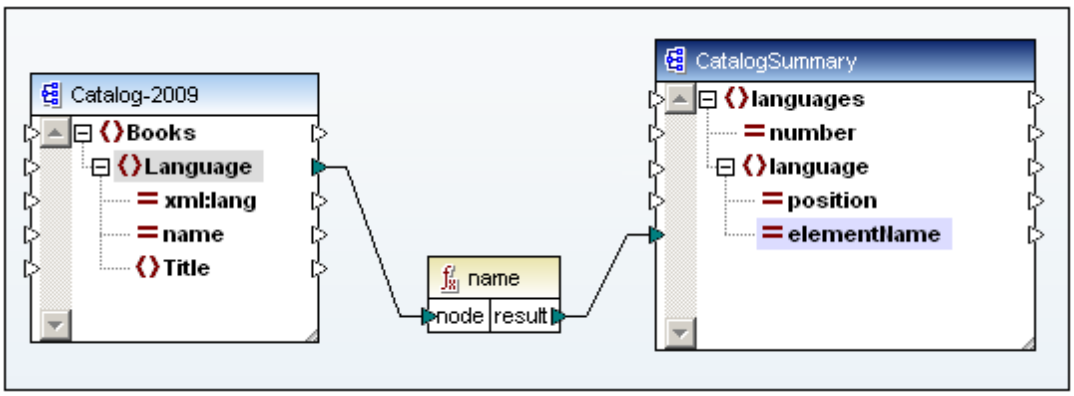
En el ejemplo, la función **`last`** devuelve la posición del último nodo del conjunto de nodos de contexto (el conjunto de nodos de los elementos `Language`) como valor del atributo `number`. Este valor también es el tamaño del conjunto de nodos, puesto que indica el número de nodos que contiene el conjunto.

La función **`position`** devuelve la posición del nodo `Language` que se está procesando en ese momento. La posición de cada nodo `Language` dentro del conjunto de nodos de los elementos `Language` se escribe en el atributo `language/@position`. Recomendamos utilizar las funciones **`position`** y **`count`** de la biblioteca de funciones **`core`**.

name, local-name, namespace-uri

Estas funciones devuelven respectivamente el nombre, el nombre local y el URI de espacio de nombres del nodo de entrada. En la imagen siguiente puede ver cómo se usan (observe que no se especifica el nodo de contexto).

La función **`name`** devuelve el nombre del nodo `Language` y lo envía al atributo `language/@elementname`. Si el argumento de la función es un conjunto de nodos en lugar de un solo nodo, devuelve el nombre (o nombre local o URI de espacio de nombres) del primer nodo del conjunto de nodos.



La función **name** devuelve el QName del nodo. La función **local-name** devuelve la parte de nombre local del QName del nodo. Por ejemplo, si el QName de un nodo es `altova:MiNodo`, el nombre local es `MiNodo`.

El URI de espacio de nombres es el URI del espacio de nombres al que pertenece el nodo. Por ejemplo, se puede declarar el prefijo `altova:` para crear una asignación a un URI de espacio de nombres de esta manera: `xmlns:altova="https://www.altova.com/namespaces"`.

Nota: en la biblioteca de funciones **core** puede encontrar más funciones XPath 1.0.

7.5.20 xslt | xslt functions

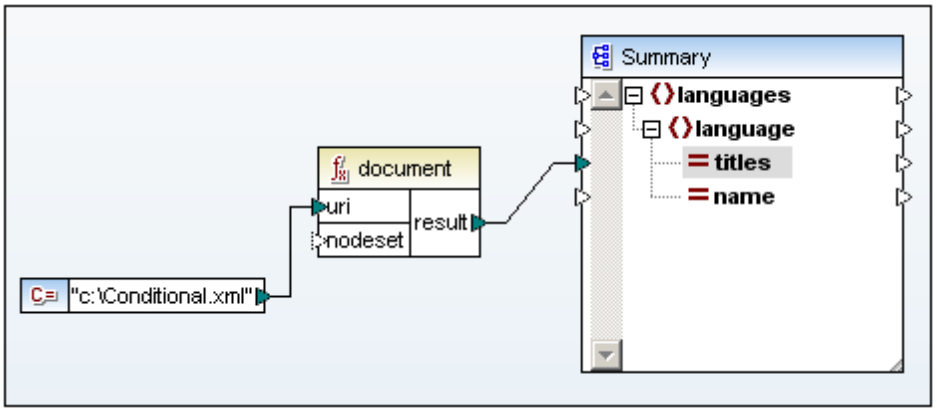
Las funciones de la biblioteca **xslt | xslt** son funciones XSLT 1.0.

7.5.20.1 currrent

Esta función no toma argumentos y devuelve el nodo actual.

7.5.20.2 document

Esta función se aplica a un documento XML externo (con el argumento **uri**). El argumento opcional **nodeset** especifica un nodo, cuyo URI base se utiliza para resolver el URI dado en el primer argumento si este URI es relativo. El resultado de la función se envía a un nodo del documento de destino.



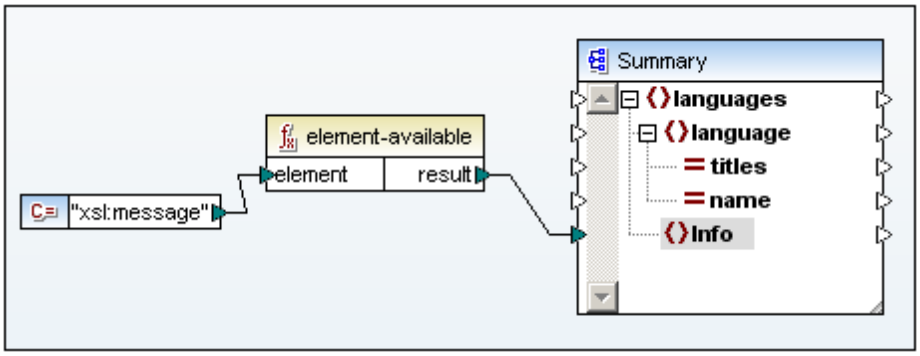
No olvide que el argumento **uri** es una cadena y debe ser la ruta de acceso absoluta de un archivo.

7.5.20.3 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.

7.5.20.4 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.

7.5.20.5 generate-id

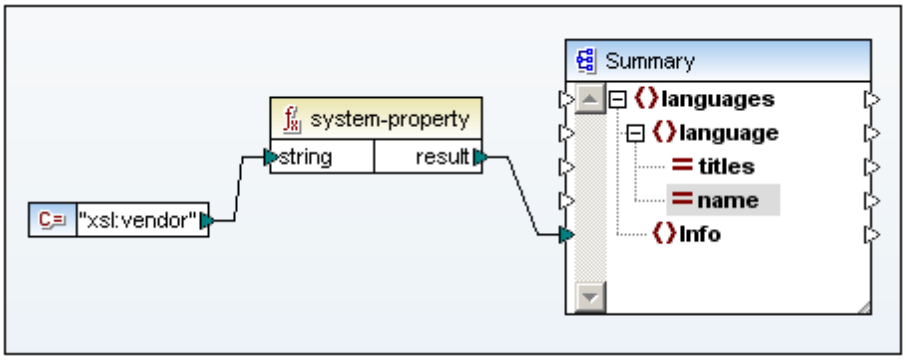
Esta función genera una cadena única que identifica el primer nodo del conjunto de nodos identificado por el argumento de entrada opcional.

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.

7.5.20.6 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:prefix`, porque este es el prefijo asociado con el espacio de nombres XSLT de la hoja de estilos XSLT subyacente.



7.5.20.7 unparsed-enity-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).

Altova MapForce 2018 Basic Edition

Automatizar asignaciones de datos

8 Automatizar asignaciones de datos

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 o XQuery (*véase Automatización con RaptorXML Server*).
- *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 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 (https://www.altova.com/es/download-trial-server.html)::

- RaptorXML Server es un motor de procesamiento XML ultrarápido compatible con XML, XML Schema, XSLT, XPath, XQuery y muchos otros formatos. Esta edición viene con el paquete de instalación de FlowForce Server.
- 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 1.0 o 2.0 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 con el motor integrado.

8.2 Interfaz de la línea de comandos de MapForce

En este momento no podemos ofrecer este contenido en su idioma, pero aquí tiene la versión en inglés.

The general syntax of a MapForce command at the command line is:

```
MapForce.exe <filename> [/{target} [[<outputdir>] [/options]]]
```

Legend

The following notation is used to indicate command line syntax:

Notation	Description
Text without brackets or braces	Items you must type as shown
<Text inside angle brackets>	Placeholder for which you must supply a value
[Text inside square brackets]	Optional items
{Text inside braces}	Set of required items; choose one
Vertical bar (|)	Separator for mutually exclusive items; choose one
Ellipsis (...)	Items that can be repeated

<filename>

The mapping design (.mfd) file from which code is to be generated.

/{target}

Specifies the target language or environment for which code is to be generated. The following code generation targets are supported.

Target	Description
/XSLT	Generates XSLT 1.0 code.
/XSLT2	Generates XSLT 2.0 code.

<outputdir>

Optional parameter which specifies the output directory. If an output path is not supplied, the current working directory will be used. Note that any relative file paths are relative to the current working directory.

/options

The `/options` are not mutually exclusive. One or more of the following options can be set.

Option	Description
`/GLOBALRESOURCEFILE <filename>`	This option is applicable if the mapping uses Global Resources to resolve input or output file or folder paths, or databases. For more information, see Altova Global Resources. The option `/GLOBALRESOURCEFILE` specifies the path to a Global Resource .xml file. Note that, if `/GLOBALRESOURCEFILE` is set, then `/GLOBALRESOURCECONFIG` must also be set.
`/GLOBALRESOURCECONFIG <config>`	This option specifies the name of the Global Resource configuration (see also the previous option). Note that, if `/GLOBALRESOURCEFILE` is set, then `/GLOBALRESOURCECONFIG` must also be set.
`/LOG <logfilename>`	Generates a log file at the specified path. `<logfilename>` can be a full path name, for example, it can include both a directory and a file name. However, if a full path is supplied, the directory must exist for the log file to be generated. If you only specify the file name, then the file will be placed in the `<outputdir>` directory.

Remarks

- Relative paths are relative to the working directory, which is the current directory of the application calling MapForce. This applies to the path of the .mfd filename, output directory, log filename, and global resource filename.
- Do not use the end backslash and closing quote at the command line (for example, "C:\My directory\"). These two characters are interpreted by the command line parser as a literal double quotation mark. Use the double backslash \\ if spaces occur in the command line and you need the quotes ("c:\My Directory\\"), or try to avoid using spaces and therefore quotes at all.

Examples

1) To start MapForce and open the mapping `<filename>.mfd`, use:

```
MapForce.exe <filename>.mfd
```

2) To generate XSLT 2.0 code and also create a log file with the name <logfilename>, use:

```
MapForce.exe <filename>.mfd /XSLT2 <outputdir> /LOG <logfilename>
```

3) To generate XSLT 2.0 code taking into account the global resource configuration <grconfigname> from the global resource file <grfilename>, use:

```
Mapforce.exe <filename>.mfd /XSLT2 <outputdir> /GLOBALRESOURCEFILE
<grfilename> /GLOBALRESOURCECONFIG <grconfigname>
```

Altova MapForce 2018 Basic Edition

Personalizar MapForce

9 Personalizar MapForce

Esta sección del manual explica cómo trabajar con recursos globales de Altova y con archivos de catálogo.

9.1 Cambiar opciones de MapForce

Las opciones generales de configuración de MapForce se pueden cambiar con el comando **Opciones** del menú **Herramientas**.

El cuadro de diálogo "Opciones" incluye varias pestañas:

Bibliotecas

- Desde aquí puede agregar o eliminar bibliotecas de funciones personales (*véase Importar funciones XSLT 1.0 o 2.0 personales*).

Generales

- Puede especificar si el logotipo aparece o se oculta cuando se inicia MapForce.
- Puede habilitar y deshabilitar el fondo degradado del panel *Asignación*.
- Puede activar la opción *Limitar la presentación de anotaciones...* que afecta a los componentes que admiten anotaciones (p. ej. esquemas XML, archivos EDI, etc.). Si el texto de la anotación tiene varias líneas y habilita esta opción, solamente se presentarán las primeras X líneas del componente. Esta opción también afecta a las instrucciones SELECT que estén visibles en los componentes.
- Puede definir la codificación de caracteres predeterminada para componentes nuevos. Esta configuración se puede modificar en cada componente por separado (*véase Cambiar la configuración del componente*).
- Puede definir el tiempo de espera de ejecución para la vista previa de resultados en el panel *Resultados*.
- Puede especificar si el resultado de la asignación se escribe en archivos temporales (opción predeterminada) o en los archivos finales directamente cuando se genera la vista previa de resultados en el panel *Resultados*.

  Advertencia: si habilita la opción "Escribir directamente en archivos de salida finales" sobrescribirá los archivos de salida sin solicitar la confirmación previa del usuario.

- Puede especificar el tamaño máximo del texto que aparece en el panel Resultado cuando se genera la vista previa de resultados de asignaciones que generan archivos XML y de texto de gran tamaño (*véase Vista previa de resultados*).

Edición

- Puede especificar si los componentes y funciones deben alinearse con los demás componentes cuando se arrastran con el ratón (*véase Alinear componentes*).
- Si habilita la opción *Eliminación inteligente de componentes*, la aplicación recordará las conexiones que pertenecían a componentes eliminados (*véase Conservar conexiones tras eliminación de componentes*).

Mensajes

- Aquí puede volver a habilitar notificaciones de mensajes deshabilitados previamente al activar la casilla *No volver a mostrar este mensaje*.

9.2 Recursos globales de Altova

Los recursos globales de Altova es una característica que permite hacer referencia a archivos, carpetas y bases de datos y poder reutilizar este tipo de recursos, configurarlos y ponerlos a disposición de otras aplicaciones de Altova.

Por ejemplo, imagine que varias asignaciones de MapForce leen datos del mismo archivo XML de forma rutinaria. Si por algún motivo el nombre del archivo cambiara, se producirían errores de tipo "archivo no encontrado" en varios contextos y el flujo de trabajo quedaría dañado. Para evitar este tipo de problemas podemos crear *alias de archivo* (en otras palabras, un recurso global) y cambiar las asignaciones para que hagan referencia a este recurso global en lugar de remitir al archivo en el disco. De este modo, si el nombre de archivo llegara a cambiar, solo haría falta actualizar el alias de archivo.

Puede definir recursos globales y compartirlos en todas estas aplicaciones de escritorio de Altova: Authentic, MobileTogether Designer, MapForce, DatabaseSpy y XMLSpy. En el lado servidor, estas aplicaciones servidor de Altova pueden consumir recursos globales: MapForce Server, MapForce Server Advanced Edition, RaptorXML Server, RaptorXML+XBRL Server.

Los recursos globales (tanto si se trata de archivos, de carpetas o de bases de datos) se pueden usar en MapForce en numerosos casos:

- Para suministrar rutas de acceso configurables como entrada de la asignación (*véase* *Ejemplo: ejecutar asignación con archivos de entrada variables*).
- Para redirigir el resultado de la asignación a una ruta de acceso configurable (*véase* *Ejemplo: generar resultados en carpetas variables*).

Nota

- FlowForce Server no admite el uso de recursos globales. MapForce Server puede consumir recursos globales en la línea de comandos o a nivel de API.
- MapForce Basic Edition no puede consumir conexiones de base de datos definidas como recursos globales.

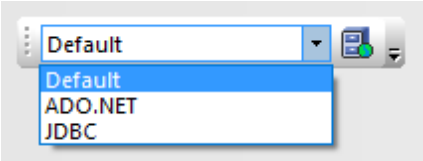
9.2.1 Crear recursos globales

En este momento no podemos ofrecer este contenido en su idioma, pero aquí tiene la versión en inglés.

A Global Resource alias is a reusable reference which represents a file or folder path, or a database connection. Aliases are defined only once and can be reused as many times as necessary in contexts which support them, including across multiple Altova applications. Taking databases as example, if you frequently work with a specific database in more than one Altova application, then it is a good idea to add the database connection as a Global Resource. This

way, you wouldn't need to go through all the Database Connection Wizard steps each time when you need to connect to the same database from another Altova application.

File, folder, and database aliases are configurable in their turn, by means of so-called "configurations". Configurations make it possible to easily switch between files, folders and databases that are consumed or produced by Altova applications, which is particularly useful for testing scenarios. For example, you could create a database alias that consists of three separate connections to the same database, each with a different driver kind: (a) ODBC, the default connection kind, (b) JDBC, and (c) ADO.NET. This way, to connect to the database with a specific driver, you would just select the corresponding configuration from the Global Resources drop-down list before running the mapping.



Global Resources drop-down list

Configurations can also help you generate mapping output to variable folders, with a click of a button. For example, you could create a folder alias with two configurations: (a) "Testing", which points to directory **C:\Testing** and (b) "Production", which points to directory **C:\Production**. It is then possible to configure a mapping to generate output to either **C:\Testing** or **C:\Production** folders, just by selecting the required configuration from the Global Resources drop-down list before running the mapping. This example is discussed in more detail in Example: Generate Output to Variable Folders.

How to create a Global Resource alias

1. On the **Tools** menu, click **Global Resources**. (Alternatively, click the **Global Resource** toolbar button.)
2. Click **Add** and select the resource type you wish to create (file, folder, database).
3. Enter a descriptive name for this alias in the **Resource alias** text box (for example, "MappingInputFile", "MappingOutputFolder", "DatabaseConnection").
4. Set up the "Default" configuration:
   a) If it's a file or folder, browse for the file or folder to which this resource should point by default.
   b) If it's 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 the mapping runs (unless a different configuration is explicitly selected from the Global Resources drop-down list or supplied as a command line parameter in server execution).
5. Optionally, if the resource should have an additional configuration (for example, a driver kind in case of databases, or an alternative path in case of files or folders), click the **Add configuration** button, enter a descriptive name (for example "ProductionFolder" or "JDBC_Alternative"), and set it up as follows:
   a) If it's a file or folder, browse for the file or folder to which this resource should point as an alternative to the default configuration defined in previous step.
   b) If it's a database connection, follow the Database Connection Wizard to connect to the database. This database connection will be used as an alternative to the default one.

In some cases, it might be more convenient to create a configuration as a copy of the default configuration, and then edit it. In this case, click the **Add configuration as a copy of the currently selected configuration** button.

6. Repeat the previous step for each additional configuration required.

9.2.2 Archivo XML de recursos globales

Todos los recursos globales de Altova, independientemente de la aplicación en la que se crearan, se almacenan por defecto en esta ruta de acceso: **C:\Usuarios\Documentos\Altova\GlobalResources.xml**. Esto garantiza la transparencia del proceso, permite crear copias de seguridad con facilidad y facilita su transporte a otros equipos donde estén instalados los productos de Altova. También puede cambiar el nombre del archivo **GlobalResources.xml** o copiarlo para crear varios archivos de recursos globales. No obstante, cada aplicación de Altova puede tener como máximo un archivo de recursos globales activo.

Para configurar el archivo de recursos globales activo:

1. En el menú **Herramientas** haga clic en **Recursos globales** (o en el botón **Recurso global** de la barra de herramientas.)
2. Haga clic en **Examinar** y seleccione el archivo XML de recursos globales necesario.

Si usa varios archivos de recursos globales, asegúrese de que el archivo que está activo contiene todos los recursos globales que necesita para ejecutar la asignación. Por ejemplo, si la asignación se configuró para leer datos de una ruta de acceso con ayuda de un recurso global, el archivo activo de recursos globales debe contener dicho recurso global. De lo contrario, recibirá mensajes de error de tipo "Error al resolver el recurso global" en la ventana Mensajes.

9.2.3 Ejemplo: ejecutar asignación con archivos de entrada variables

En este momento no podemos ofrecer este contenido en su idioma, pero aquí tiene la versión en inglés.

Let's assume that, as part of your job duties, you frequently run a mapping that takes as input an XML file. Under normal circumstances, whenever you want to change the input XML of the mapping, you can open the properties of the source XML component and browse for the new input file, see Changing the Component Settings. This is easy to accomplish if it's a one time task. However, what if you need to change the input XML file of the mapping multiple times per day, or even per hour? For example, every morning you need to run the mapping and generate a report by using one XML file as mapping input, and every evening the same report must be generated from another XML file. This is where Global Resources can help you: instead of editing the mapping multiple times per day (or keeping multiple copies of it), you could configure the mapping to read from a file defined as a global resource (a so-called "file alias"). To address the requirement laid

out in this example, the file alias could be configured to have two configurations:

1. "Default" - This configuration would supply a "morning" XML file as mapping input
2. "EveningReports" - This configuration would supply an "evening" XML file as mapping input.

Having these configurations in place would make it possible to run the mapping with either input file. Once the file alias is set up as shown below, you will be able to select the desired configuration from a drop-down list, before running the mapping.

Step 1: Create the Global Resource

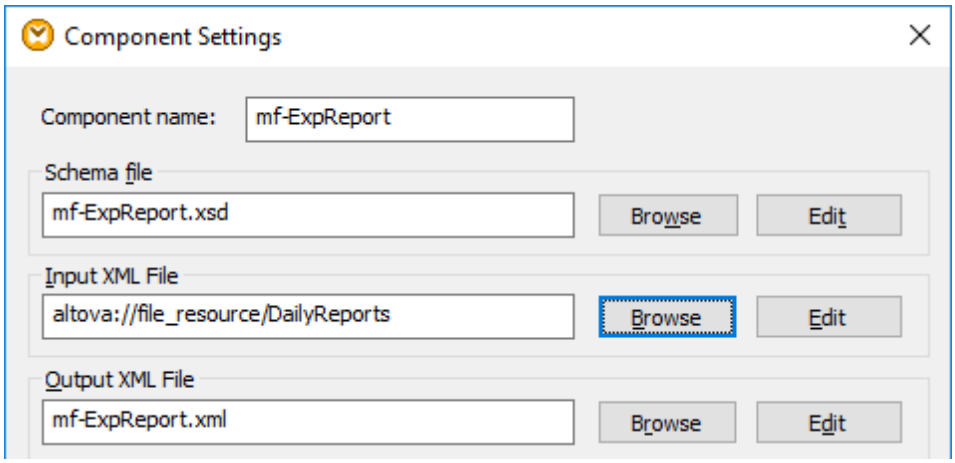
The file alias can be created as follows:

1. On the **Tools** menu, click **Global Resources**. (Alternatively, click the **Global Resource** toolbar button.)
2. Click **Add | File**.
3. Enter a name in the **Resource alias** text box (in this example, "DailyReports" would be an appropriate name).
4. Click **Browse** and select the following file: **<Documentos>\Altova\MapForce2018\MapForceExamples\Tutorial\mf-ExpReport.xml**.
5. Click **Add Configuration** and name it "EveningReports".
6. Click **Browse** and this time select the following file: **<Documentos>\Altova\MapForce2018\MapForceExamples\Tutorial\mf-ExpReport2.xml**.

Step 2: Use the Global Resource in the mapping

The required Global Resource has now been created; however, the mapping is not using it yet. To change the mapping so that it reads from the previously defined file alias (Global Resource), do the following:

1. Open the following mapping **<Documentos>\Altova\MapForce2018\MapForceExamples\Tutorial\Tut-ExpReport.mfd**.
2. Right-click the header of the source component on the mapping, and select **Properties** from the context menu.
3. Next to **Input XML file**, click **Browse**.
4. Click **Switch to Global Resources** and select the file alias "DailyReports" defined previously.
5. Click **Open**. The input XML file path has now become **altova://file_resource/DailyReports**, which indicates that the path uses a Global Resource.



Step 3: Run the mapping with the desired configuration

You can now easily switch the input XML file before running the mapping, as follows:

- On the **Tools** menu, click **Active Configuration | Default**, to use the file **mf-ExpReport.xml** as input.
- On the **Tools** menu, click **Active Configuration | EveningReports**, to use the file **mf-ExpReport2.xml** as input.

Alternatively, select the required configuration from the **Global Resources** drop-down list.

To preview the mapping result with either configuration, click the **Output** tab and observe differences in the generated output.

9.2.4 Ejemplo: generar resultados en carpetas variables

En este momento no podemos ofrecer este contenido en su idioma, pero aquí tiene la versión en inglés.

This example illustrates how mapping output can be redirected to different folders by means of Global Resources.

Let's suppose that sometimes you need to generate the mapping output to one directory (for example, **C:\Testing**), while in certain cases output must be generated to another directory (for example, **C:\Production**). With Global Resources, this is possible by creating a folder alias with two configurations:

1. "Default" configuration - Generates output to **C:\Testing**
2. "Production" configuration - Generates output to **C:\Production**.

The steps below illustrate how to achieve this goal.

Step 1: Create the Global Resource

The folder alias can be created as follows:

1. On the **Tools** menu, click **Global Resources**. (Alternatively, click the **Global Resource** toolbar button.)
2. Click **Add | Folder**.
3. Enter a name in the **Resource alias** text box (in this example, "OutputDirectory" could be an appropriate name).
4. Click **Browse** and select the following folder: **C:\Testing**. (Make sure that this folder already exists on your operating system.)
5. Click **Add Configuration** and enter a name for the new configuration (in this example, "ProductionDirectory").
6. Click **Browse** and this time select the following folder: **C:\Production**. (Make sure that this folder already exists on your operating system.)

Step 2: Use the Global Resource in the mapping

The required Global Resource has now been created; however, the mapping is not using it yet. To change the mapping so that it uses from the previously defined folder alias (Global Resource), do the following:

1. Open the following mapping **<Documentos>\Altova\MapForce2018\MapForceExamples\Tutorial\Tut-ExpReport.mfd**.
2. Right-click the target component on the mapping, and select **Properties** from the context menu.
3. Next to **Output XML file**, click **Browse**.
4. Click **Switch to Global Resources**, and then click **Save**.
5. When prompted to save the output XML file, enter **output.xml** (or another descriptive file name that you wish to give to the output file). The output XML file path has now become **altova://folder_resource/OutputDirectory/output.xml**, which indicates that the path is defined as a Global Resource.

Step 3: Run the mapping with the desired configuration

You can now easily switch to the desired mapping output folder file before running the mapping, as follows:

- On the **Tools** menu, click **Active Configuration | Default**, and then click the **Output** tab to preview the mapping result. The mapping output (either a temporary or a permanent file, as explained below) will be generated in the **C:\Testing** directory.
- On the **Tools** menu, click **Active Configuration | ProductionDirectory**, and then click the **Output** tab. The mapping output (either a temporary or a permanent file, as explained below) will be generated in the **C:\Production** directory.

Note: The mapping output is written by default as a temporary file, unless you explicitly configured MapForce to write output to permanent files.

To configure MapForce to generate permanent files instead of temporary, do the following:

1. On the **Tools** menu, click **Options**.
2. In the **General** section, select the option **Write directly to final output files**.

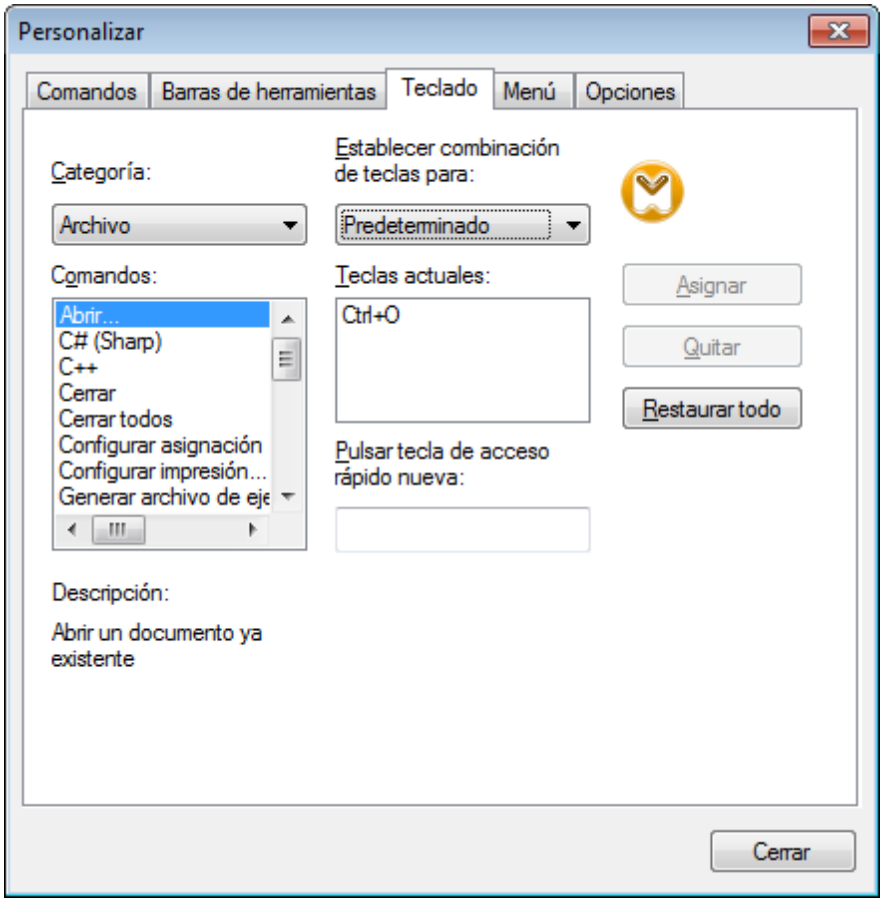
9.3 Personalizar teclas de acceso rápido

En MapForce las teclas de acceso rápido se definen y modifican así:

1. Haga clic en el menú **Herramientas** y después en **Personalizar**.
2. 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".
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 pulse las teclas que deben activar el comando.



La tecla de acceso rápido aparece inmediatamente en el cuadro de texto. Si la tecla de acceso ya está asignada a algún comando, esto se indica bajo el cuadro de texto.

5. 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 quitar la asignación de una tecla de acceso rápido o eliminarla:

1. En el cuadro de lista *Teclas actuales* haga clic en la tecla de acceso rápido que desea eliminar.
2. Haga clic en el botón **Quitar**.
3. Haga clic en **Cerrar** para confirmar.

Nota: el cuadro combinado *Establecer combinación de teclas para:* no tiene por ahora ninguna función.

Estas son las teclas de acceso rápido asignadas por defecto:

Teclas de acceso rápido	
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	Cambia de una asignación a otra
CTRL + F6	Recorre todas las ventanas abiertas
CTRL + F4	Cierra la asignación activa
Alt + F4	Cierra MapForce
Alt + F, F, 1	Abre el último archivo
Alt + F, T, 1	Abre 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	Confirma una selección

Teclas de acceso rápido del panel Resultados

CTRL + F2	Insertar o quitar marcador
F2	Siguiente marcador
Mayús + F2	Marcador anterior
CTRL + Mayús + F2	Quitar todos los marcadores

Teclas de acceso rápido de zoom

CTRL + rueda del ratón hacia adelante	Acercarse
CTRL + rueda del ratón hacia detrás	Alejarse
CTRL + 0 (cero)	Restablecer el nivel de zoom

9.4 Archivos de catálogo

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.

RootCatalog.xml

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`. MapForce busca cada uno de estos archivos de catálogo y sus URI se resuelven de acuerdo con las asignaciones especificadas en ellos.

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

En el fragmento de código anterior, observe que en las carpetas `Schemas` y `XBRL` de la carpeta identificada con la variable `%AltovaCommonFolder%` están los archivo de catálogo llamados `catalog.xml`. (El valor de la variable `%AltovaCommonFolder%` se indica en la tabla que aparece más abajo.)

Los archivos de catálogo de la carpeta de archivos comunes de Altova realizan asignaciones entre los identificadores de sistema e identificadores públicos predefinidos de las taxonomías XBRL y los esquemas (como SVG y WSDL) y los identificadores URI que apuntan a copias locales de los respectivos esquemas. Estos esquemas se instalan en la carpeta de archivos comunes de Altova cuando se instala MapForce. Rogamos no cree asignaciones duplicadas en estos archivos porque se pueden producir errores.

CoreCatalog.xml, CustomCatalog.xml y Catalog.xml

En el ejemplo de código anterior del archivo `RootCatalog.xml` se indica que se busquen los archivos `CoreCatalog.xml` y `CustomCatalog.xml`:

- `CoreCatalog.xml` contiene ciertas asignaciones propias de Altova que sirven para localizar esquemas en la carpeta de archivos comunes de Altova.
- `CustomCatalog.xml` es un archivo base en el que el usuario puede crear asignaciones propias. En el archivo `CustomCatalog.xml` puede crear asignaciones para cualquier esquema, siempre y cuando el esquema no esté controlado por los archivos de catálogo de la carpeta de archivos comunes de Altova. Para crear asignaciones use los elementos compatibles con el mecanismo de catalogación OASIS (*véase más adelante*).
- En la carpeta de archivos comunes de Altova hay varios archivos `Catalog.xml`. Cada archivo está dentro de la carpeta de un esquema o de una taxonomía XBRL concretos de la carpeta de archivos comunes de Altova. Además, cada archivo `Catalog.xml` realiza asignaciones entre los identificadores de sistema e identificadores públicos y los identificadores URI que apuntan a las copias locales de los respectivos esquemas.

Ubicación de los archivos de catálogo y los esquemas

Los archivos `RootCatalog.xml` y `CoreCatalog.xml` se instalan en la carpeta de aplicación de MapForce. El archivo `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 las carpetas: `%AltovaCommonFolder%\Schemas` y `%AltovaCommonFolder%\XBRL`.

Variables de entorno Shell y variables de Altova

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:

`%AltovaCommonFolder%`	C:\Archivos de programa\Altova\Common2018
`%DesktopFolder%`	Ruta de acceso completa de la carpeta Escritorio del usuario actual.
`%ProgramMenuFolder%`	Ruta de acceso completa de la carpeta del menú Programas del usuario actual.
`%StartMenuFolder%`	Ruta de acceso completa de la carpeta del menú Inicio del usuario actual.
`%StartUpFolder%`	Ruta de acceso completa de la carpeta Inicio del usuario actual.
`%TemplateFolder%`	Ruta de acceso completa de la carpeta de plantillas del usuario actual.
`%AdminToolsFolder%`	Ruta de acceso completa del directorio del sistema de archivos que almacena las herramientas administrativas del usuario actual.
`%AppDataFolder`	Ruta de acceso completa de la carpeta Datos de programa del usuario

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

Cómo funcionan los catálogos: documentos DTD

Los catálogos se suelen usar para redireccionar una llamada a una DTD hasta un URI local. Para ello es necesario realizar asignaciones, en el archivo de catálogo, entre los identificadores de sistema o públicos y el URI local pertinente. 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 apunte 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>
```

En este caso se leería el documento y se buscaría el identificador `PUBLIC` en el catálogo. 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, se encuentra un identificador `PUBLIC`, de modo que la búsqueda de la DTD del SVG se redirecciona al URI `schemas/svg/svg11.dtd` (esta ruta es relativa al archivo de catálogo) y este archivo local se usará como DTD. Si en el catálogo no hay una asignación para el identificador `Public`, entonces se usa la URL del documento XML (en el ejemplo anterior: `http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd`).

El subconjunto de catálogos compatible con MapForce

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. Recuerde que todos los elementos puede tomar 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"/>`

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`. Además, 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.

Extensiones de archivo y edición inteligente basada en un esquema

Mediante los archivos de catálogo también puede especificar a qué tipo de documentos (según su extensión de archivo) se les puede aplicar las funciones de edición inteligente de MapForce de acuerdo con las reglas del esquema que usted indique. Por ejemplo, si crea la extensión de archivo personalizada `.myhtml` para archivos (HTML) que deben ser válidos con respecto a la DTD HTML, entonces puede habilitar la función de edición inteligente para los archivos que tengan esta extensión. Para ello, añada el elemento que aparece a continuación al archivo de catálogo `CustomCatalog.xml` y como elemento secundario del elemento `<catalog>`:

```
<spy:fileExtHelper ext="myhtml" uri="schemas/xhtml/xhtml1-transitional.dtd"/>
```

Así se habilita la función de edición inteligente (finalización automática, ayudantes de entrada, etc.) para archivos `.myhtml` en MapForce en base a la DTD transicional XHTML 1.0. Consulte el archivo `catalog.xml` de la carpeta `%AltovaCommonFolder%\Schemas\xhtml`, que contiene entradas similares a este ejemplo.

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 `%AltovaCommonFolder%\Schemas\schema` contiene referencias a documentos 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.

Más información

Para más información, consulte la especificación XML Catalogs.

Altova MapForce 2018 Basic Edition

Referencia de menús

10 Referencia de menús

Esta sección describe los comandos de menú de MapForce.

10.1 Archivo

Nuevo
Crea un documento de asignación nuevo.

Abrir
Abre archivos de asignación (*.mfd) . Tenga en cuenta que no se pueden abrir asignaciones con características de una edición superior de MapForce.

Guardar
Guarda la asignación activa usando el nombre de archivo activo en ese momento.

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.

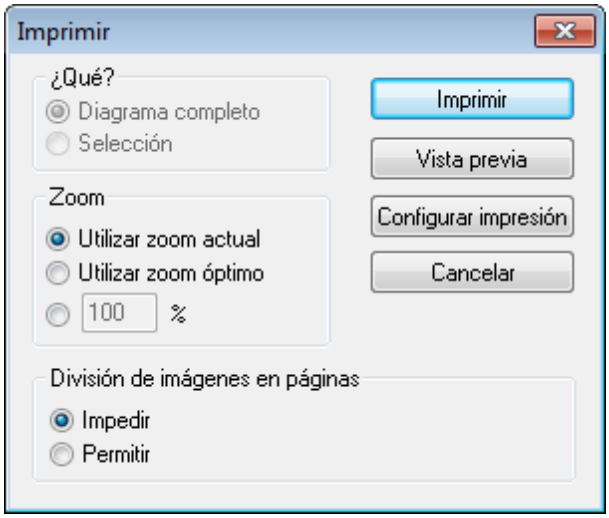
Guardar todos
Guarda todos los archivos que están abiertos.

Volver a cargar
Vuelve a cargar el archivo de asignación activo. Aparece un aviso preguntando si desea descartar los cambios realizados o guardarlos.

Cerrar
Cierra el archivo de asignación activo. Aparece un aviso preguntando si desea descartar los cambios realizados o guardarlos.

Cerrar todos
Cierra todos los archivos de asignación activos. Aparece un aviso preguntando si desea guardar los archivos que están sin guardar.

Imprimir
Abre el cuadro de diálogo "Imprimir".



Cuadro de diálogo "Imprimir"

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.

Vista previa de impresión
Abre el cuadro de diálogo "Imprimir" (ver comando **Imprimir**).

Configurar impresión
Abre el cuadro de diálogo "Configurar impresión" donde puede definir la impresora y la configuración del papel.

Validar asignación
Comprueba si todas las asignaciones (conectores) son válidas y emite advertencias y errores de validación (*véase* *Validar asignaciones*).

Configurar asignación
Abre el cuadro de diálogo donde puede definir las opciones de configuración del documento activo (*véase* *Cambiar configuración de la asignación*).

Generar código en el lenguaje seleccionado
Genera código en el lenguaje seleccionado para la asignación activa. El lenguaje que está seleccionado está visible en la barra de herramientas: XSLT, XSLT 2.

Generar código en | XSLT (XSLT2)
Genera los archivos XSLT necesarios para la transformación de datos a partir de los archivos de origen. Al hacer clic en este comando se abre el cuadro de diálogo "Buscar carpeta" donde puede seleccionar la ubicación de los archivos XSLT. El nombre de los archivos XSLT que se generan se define en el cuadro de diálogo "Configurar asignación" (*véase* *Cambiar configuración de la asignación*).

Archivos recientes
Una lista con archivos abiertos recientemente.

Salir
Cierra la aplicación. Antes aparece un aviso preguntando si desea guardar los archivos que estén sin guardar.

10.2 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 la pestaña *XSLT*.

Deshacer 

MapForce ofrece un número ilimitado de operaciones deshacer, que puede usar para retroceder en todos los cambios realizados.

Rehacer 

Este comando permite rehacer cambios anulados anteriormente con el comando **Deshacer**. Puede avanzar y retroceder por el historial de cambios usando estos dos comandos.

Buscar 

Permite buscar una cadena de texto en las pestañas *XSLT*, *XSLT2* o *Resultados*.

Buscar siguiente (F3) 

Busca la siguiente instancia del término de búsqueda.

Buscar anterior (Mayús F3) 

Busca la instancia anterior del término de búsqueda.

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* o *Resultados*.

10.3 Insertar

Archivo o esquema XML 

Inserta un archivo de esquema XML o de instancia XML a la asignación. Si selecciona un archivo XML que haga referencia a un esquema, la asignación no necesita más información. Si selecciona un archivo XML sin referencia a un esquema, deberá generar automáticamente el esquema XML correspondiente (*véase* *Generar un esquema XML*). 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.

Insertar componente de entrada 

Si en el área de asignación está visible una asignación de datos, este comando inserta un componente de entrada (*véase* *Pasar parámetros a la asignación*). 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* *Definir componentes de entrada complejos*).

Insertar componente de salida 

Si en el área de asignación está visible una asignación de datos, este comando inserta un componente de salida (*véase* *Obtener valores de cadena de una asignación*). Si en el área de asignación está visible una función definida por el usuario, este comando agrega un componente de salida en la función (*véase* *Definir componentes de salida complejos*).

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.

Variable 

Inserta una variable 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 (*véase* *Variables intermedias*).

Componente de ordenación: nodos/filas

Inserta un componente que permite ordenar nodos (*véase* *Ordenar datos*).

Filtro: nodos/filas 

Inserta un componente que usa dos parámetros de entrada (**`node/row`** y **`bool`**) y dos de salida (**`on-true`** y **`on-false`**). Si el valor booleano es `true`, el valor del parámetro **`node/row`** se pasa al parámetro **`on-true`**. Si el booleano es `false`, el valor del parámetro **`node/row`** se pasa al parámetro **`on-false`**. Consulte la sección Filtros y condiciones para obtener más información.

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). Consulte el apartado Usar asignaciones de valores para obtener más información.

Condición IF-Else 

Inserta un componente de tipo If-Else (véase Filtros y condiciones).

10.4 Componente

Cambiar de elemento raíz
Permite cambiar el elemento raíz del documento de instancia XML.

Editar la definición del esquema en XMLSpy
Seleccione esta opción (después de hacer clic en el documento/esquema XML) para abrir el archivo de esquema XML en la vista Esquema de XMLSpy.

Agregar delante un duplicado de entrada
Inserta una copia (un clon) del elemento seleccionado antes del elemento seleccionado. Los elementos duplicados no tienen iconos de salida, es decir, no se pueden usar como origen de datos. Para ver un ejemplo consulte la sección Asignar varios orígenes de datos a un destino del Tutorial. Haga clic con el botón derecho en un duplicado para moverlo a una posición nueva (opciones **Subir/Bajar** del menú contextual que aparece).

Agregar detrás un duplicado de entrada
Inserta una copia (un clon) del elemento seleccionado después del elemento seleccionado. Los elementos duplicados no tienen iconos de salida, es decir, no se pueden usar como origen de datos. Para ver un ejemplo consulte la sección Asignar varios orígenes de datos a un destino del Tutorial. Haga clic con el botón derecho en un duplicado para moverlo a una posición nueva (opciones **Subir/Bajar** del menú contextual que aparece).

Quitar el duplicado
Quita el elemento duplicado seleccionado. Para ver un ejemplo consulte la sección Asignar varios orígenes de datos a un destino del Tutorial.

Alinear el árbol a la izquierda
Coloca todos los elementos de un componente en el lado izquierdo de la ventana.

Alinear el árbol a la derecha
Coloca todos los elementos de un componente en el lado derecho de la ventana. Este diseño es práctico para crear asignaciones de datos cuyo destino es un esquema.

Propiedades
Abre un cuadro de diálogo con las opciones de configuración actuales del componente (*véase Cambiar configuración de los componentes*).

10.5 Conexión

Conectar automáticamente los secundarios equivalentes
Activa o desactiva la opción **Conexión automática de secundarios** así como el icono de la barra de herramientas.

Configurar la conexión de secundarios equivalentes
Abre el cuadro de diálogo "Configurar la conexión de secundarios equivalentes" donde puede definir las opciones de conexión (*véase* *Conectar elementos secundarios equivalentes*).

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. Aparece un cuadro de diálogo donde puede definir la conexión. Las opciones definidas en este cuadro de diálogo se conservan y se aplican cada vez que se conecten dos elementos siempre y cuando esté activo el icono **Conexión automática de secundarios**  de la barra de herramientas (*véase* *Conectar elementos secundarios equivalentes*).

Basada en el destino (estándar)
Cambia el tipo de conector a una asignación estándar (*véase* *Conexiones basadas en el destino*).

Copia total (copia los elementos secundarios)
Crea conectores para todos los elementos secundarios equivalentes, donde cada uno de los conectores secundarios aparece como una rama del conector primario (*véase* *Conexiones de copia total*).

Basada en origen (contenido mixto)
Cambia el tipo de conector a una asignación basada en origen / de contenido mixto (*véase* *Conexiones basadas en el origen*).

Propiedades
Abre un cuadro de diálogo donde puede definir la configuración específica (contenido mixto) del conector actual. Las opciones que no estén disponibles en ese momento aparecen atenuadas. No olvide que estas opciones de configuración también afectan a los elementos `complexType`, que no tienen nodos de texto (*véase* *Configuración de las conexiones*).

10.6 Función

Crear una función definida por el usuario
Crea una función nueva definida por el usuario (*véase* *Funciones definidas por el usuario*).

Crear 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.

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.

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*). 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* *Definir componentes de entrada complejos*).

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*). Si en el área de asignación está visible una función definida por el usuario, este comando agrega un componente de salida en la función (*véase* *Definir componentes de salida complejos*).

10.7 Resultados

XSLT 1.0, XSLT 2.0, XQuery, Java, C#, C++, Motor de ejecución integrado

Establece el lenguaje de transformación en el que se debe ejecutar la asignación de datos (*véase Seleccionar un lenguaje de transformación*).

Validar archivo de salida

Valida el archivo XML de salida con el esquema al que se hace referencia (*véase Validar el resultado de la asignación*).

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 (*véase Procesar varios archivos de entrada o salida de forma dinámica*).

Volver a generar archivo de salida

Vuelve a generar los datos visibles en el panel *Resultados*.

Insertar o quitar marcador

Inserta un marcador en la posición del cursor en el panel *Resultados*.

Marcador siguiente

Navega hasta el siguiente marcador en el panel *Resultados*.

Marcador anterior

Navega hasta el marcador anterior en el panel *Resultados*.

Quitar todos los marcadores

Quita todos los marcadores que están definidos en el panel *Resultados*.

Texto XML 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 utilice la opción *Tamaño de la tabulación* del grupo de opciones *Tabulaciones*.

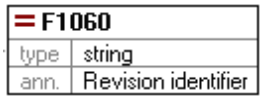
Configurar la vista Texto

Abre el cuadro de diálogo "Configurar la vista Texto", donde puede personalizar la configuración de esta vista en los paneles *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.

10.8 Vista

Mostrar anotaciones

Muestra las anotaciones del esquema XML en la ventana del componente.
Si también se activa el icono **Mostrar tipos**, el tipo y la anotación aparecen en una tabla.



Mostrar tipos

Muestra los tipos de datos del esquema para cada elemento o atributo.
Si también se activa el icono **Mostrar anotaciones**, el tipo y la anotación aparecen en una tabla.

Mostrar biblioteca en el título de la función

Muestra el nombre de la biblioteca en paréntesis en el título de la función.

Mostrar información rápida

Muestra información rápida con un texto explicativo al pasar el puntero del ratón por encima de la función.

Mostrar los conectores del componente seleccionado

Permite alternar entre una vista con todos los conectores de la asignación y una vista con los conectores relacionados con los componentes que están seleccionados.

Mostrar los conectores desde su origen a su destino

Permite alternar entre:

- una vista con todos los conectores **directamente unidos** al componente que está seleccionado y
- una vista con los conectores unidos al componente seleccionado, que empiezan en el componente de origen y terminan en el de destino

Zoom

Abre el cuadro de diálogo de zoom, donde puede indicar el factor de zoom en formato numérico o con un control deslizante.

Atrás

Retrocede en las asignaciones que están abiertas en el panel **Asignación**.

Adelante

Avanza por 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.

Ventana Bibliotecas

Muestra/oculta la ventana Biblioteca, que contiene todas las funciones.

Mensajes

Muestra/oculta la ventana Mensajes, donde aparecen los resultados de la validación, etc. 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. Arrastre el rectángulo rojo por la ventana de vista general para navegar por el panel **Asignación**.

10.9 Herramientas

Recursos globales
Abre el cuadro de diálogo "Administrar recursos globales" donde puede agregar, editar o eliminar recursos globales en el archivo XML de recursos globales (*véase* *Recursos globales de Altova*).

Configuración activa
Permite seleccionar la configuración de recurso global que está activa en una lista que enumera todas las configuraciones definidas previamente como recursos globales.

Crear asignación invertida
Crea una asignación *invertida* a partir de la asignación activa que será la base de una asignación nueva. Tenga en cuenta que el resultado no es una asignación completa porque 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.

Cuando se invierte una asignación de datos, el componente de origen pasa a ser el componente de destino y viceversa. Si se asignó un archivo de instancia XML de entrada o salida a un componente, estos archivos también se permutarán.

Estos son los datos que se conservan:

- conexiones directas entre los componentes
- conexiones directas entre los componentes de una asignación en cadena
- el tipo de conexión estándar, de contenido mixto y de copia total
- la configuración de componentes de paso a través
- los componentes de base de datos

Estos son los datos que no se conservan:

- conexiones a través de funciones, filtros, etc. así como las funciones, los filtros, etc.
- funciones definidas por el usuario
- componentes de servicio web

Restaurar barras de herramientas y ventanas
Restaura las barras de herramientas, los ayudantes de entrada, las ventanas acopladas, etc. a su estado predeterminado. Es necesario reiniciar MapForce para que los cambios surtan efecto.

Personalizar...
Abre un cuadro de diálogo donde puede personalizar la interfaz gráfica del usuario de MapForce, incluidas las barras de herramientas que aparecen en la interfaz. También permite editar los menús contextuales y las teclas de acceso rápido (*véase* *Personalizar teclas de acceso rápido*).

Opciones
Abre un cuadro de diálogo donde puede cambiar la configuración predeterminada de MapForce

(*véase* *Cambiar opciones de MapForce*).

10.10 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.

1
2

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.

10.11 Ayuda

Contenido

Descripción

Abre la ayuda en pantalla de MapForce por la tabla de contenido, que aparece en el panel izquierdo de la ventana de ayuda. Esta tabla de contenido ofrece un resumen de la documentación. Haga clic en una entrada de la tabla para abrir la sección correspondiente de la documentación.

Índice

Descripción

Abre la ayuda en pantalla de MapForce por el índice de palabras clave, que aparece en el panel izquierdo de la ventana de ayuda. Este índice enumera todas las palabras clave de la ayuda y permite navegar a un tema con solo hacer doble clic en la palabra clave correspondiente. Si una palabra clave está asociada a varios temas, la ventana de ayuda muestra todos estos temas en pantalla.

Buscar

Descripción

Abre la ayuda en pantalla de MapForce por la función de búsqueda, que aparece en el panel izquierdo de la ventana de ayuda. Para buscar un término introduzca el término de búsqueda en el campo de consulta y pulse **Entrar**. El sistema de ayuda realiza una búsqueda en toda la documentación y devuelve una lista de resultados. Haga doble clic en una entrada para abrir la sección correspondiente de la documentación.

Activación del software

Descripción

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.

- ***Clave de evaluación gratuita:*** cuando inicie el software por primera vez aparecerá el cuadro de diálogo "Activación del software". Este cuadro de diálogo incluye un botón para solicitar un código clave de evaluación gratuita. Introduzca su nombre, el nombre de su compañía y su dirección de correo electrónico y haga clic en **Enviar solicitud**. Altova le enviará la clave de evaluación al correo electrónico proporcionado. Ahora introduzca el código clave en el cuadro de diálogo "Activación del software" y haga clic en **Aceptar**

para empezar a trabajar con el software, que permanecerá desbloqueado durante 30 días.

- ***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. Altova ofrece licencias para un solo usuario y licencias para varios usuarios (ambos tipos estarán en el correo electrónico que reciba de Altova). Las licencias para un solo usuario contienen los datos de la licencia, su nombre, el nombre de su compañía, su correo electrónico y un código clave. Las licencias para varios usuarios contienen los datos de la licencia, el nombre de su compañía y un código clave. Recuerde que el contrato de licencia prohíbe instalar más copias de las permitidas por la licencia adquirida. Recuerde introducir los datos que solicita el cuadro de diálogo de registro tal y como aparecen en el correo electrónico que recibió con las licencias.

Nota: recuerde que los datos de la licencia en el cuadro de diálogo "Activación del software" deben ser idénticos a los que aparecen en el correo electrónico que recibió con las licencias. En el caso de licencias para varios usuarios, cada usuario debe introducir su nombre en el campo *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:

- Los detalles de su licencia (nombre, compañía, correo electrónico, código clave)
- En un adjunto, un archivo de licencia con la extensión **`.altova_licenses`**

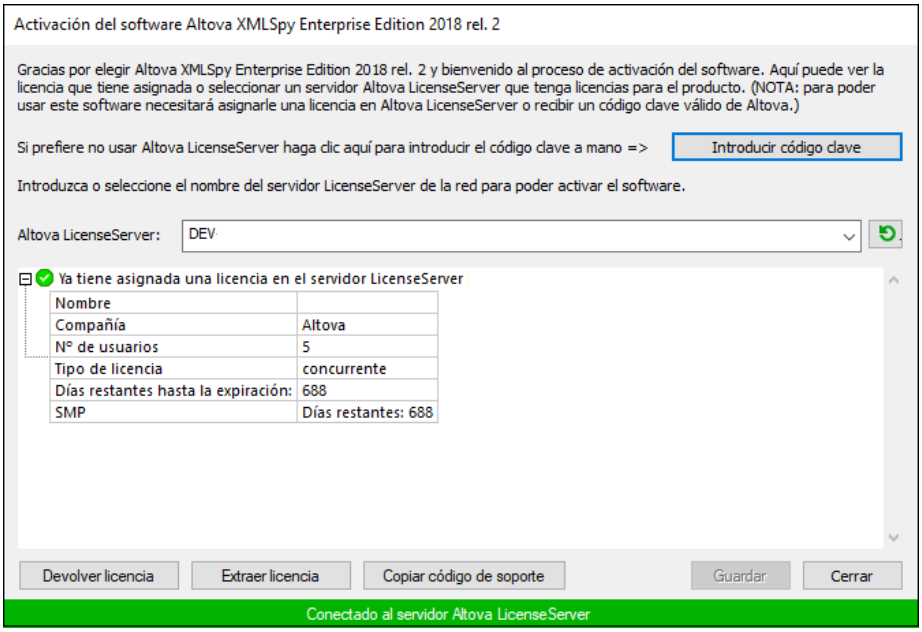
Para activar su producto de Altova, puede optar por una de las siguientes opciones:

- Introducir los detalles de la licencia recibidos con el correo electrónico en el cuadro de diálogo de activación de software de su producto de Altova y hacer clic en **Activar**.
- 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 y cargarlo desde esa ubicación a su Altova LicenseServer. Puede: (i) adquirir la licencia de su producto Altova con el cuadro de diálogo de activación de software del producto 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**.

Hay dos maneras de activar el software:

- Introduciendo la información de la licencia (con el botón **INTRODUCIR código clave...**) o
- Adquiriendo una licencia a través de un servidor Altova LicenseServer de la red (haciendo clic en el botón **Usar Altova LicenseServer**, situado al final del cuadro de diálogo). 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 (*imagen siguiente*) y basta con hacer 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 automáticamente a su estado insertado cuando finaliza el período de extracción de la licencia. 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:

1. En el cuadro de diálogo "Activación del software" haga clic en el botón **Extraer licencia** (*imagen anterior*).
2. Aparece el cuadro de diálogo "Extracción de licencias". Seleccione el periodo de extracción y haga clic en **Extraer**.
3. La licencia se extrae y 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. Ahora, en lugar del botón **Extraer licencia**, aparece el botón **Insertar licencia**. Para insertar la licencia 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.

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 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 sus características administrativas. Altova LicenseServer puede descargarse gratis del sitio web de Altova. Para más información consulte la documentación de Altova LicenseServer.

Formulario de pedido

Descripción

Hay dos maneras de comprar licencias para los productos de Altova: con el botón **Comprar código clave...** del cuadro de diálogo "Activación del software" (ver apartado anterior) o con el comando **Ayuda | Formulario de pedido,** que le lleva directamente a la tienda en línea de Altova.

Registro del software

Descripción

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

Descripción

Comprueba si existe una versión más reciente del producto en el servidor de Altova y emite un mensaje a tal efecto.

Centro de soporte técnico

Descripción

Es un enlace al centro de soporte técnico del sitio web de Altova. 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.

Preguntas más frecuentes

Descripción

Es un enlace a la página de preguntas frecuentes del sitio web de Altova. Esta página se actualiza constantemente con las preguntas que recibimos de nuestros clientes.

Descargar herramientas gratis y componentes

Descripción

Es un enlace al centro de descargas de componentes del sitio web de Altova. Aquí puede descargar 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

Descripción

Es un enlace al sitio web de Altova, donde encontrará más información sobre MapForce, otros productos de Altova y tecnologías relacionadas.

Cursos de MapForce

Descripción

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

Descripción

Abre la pantalla de presentación de la aplicación, que incluye el número de versión del producto e información sobre copyright.

Altova MapForce 2018 Basic Edition

Anexos

11 Anexos

Estos anexos contienen datos técnicos sobre MapForce e información importante sobre las licencias. Cada anexo incluye varios apartados:

Datos técnicos

- Requisitos de SO y memoria
- Analizador XML de Altova
- Motores XSLT y XQuery de Altova
- Compatibilidad con Unicode
- Uso de Internet
- Medición de licencias

Información sobre licencias

- Distribución electrónica de software
- Derechos de propiedad intelectual
- Contrato de licencia para el usuario final de Altova

11.1 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.

11.1.1 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:

- XSLT 1.0
- XSLT 2.0
- XQuery 1.0

11.1.1.1 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.

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 (` `).

11.1.1.2 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 solamente es relevante cuando se utiliza el motor XSLT 2.0 para procesar una hoja de estilos XSLT 1.0. Tenga en cuenta que los resultados obtenidos con el motor XSLT 1.0 pueden ser diferentes a los obtenidos con el motor XSLT 2.0 en modo de compatibilidad con versiones antiguas.

Espacios de nombres

En su hoja de estilos XSLT 2.0 debe declarar estos espacios de nombres para poder usar los constructores de tipo y las funciones disponibles en XSLT 2.0. Los prefijos que aparecen a continuación son los que se suelen usar, pero puede usar otros prefijos si quiere.

Espacio de nombres	Prefijo	URI del espacio de nombres
Tipos XML Schema	`xs:`	`http://www.w3.org/2001/XMLSchema`
Funciones XPath 2.0	`fn:`	`http://www.w3.org/2005/xpath-functions`

Estos espacios de nombres se suelen declarar en el elemento `xsl:stylesheet` o en el elemento `xsl:transform`:

```
<xsl:stylesheet version="2.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:fn="http://www.w3.org/2005/xpath-functions"
    ...
</xsl:stylesheet>
```

Es necesario tener en cuenta que:

- El motor XSLT 2.0 utiliza el espacio de nombres Funciones XPath 2.0 y XQuery 1.0

como **espacio de nombres de funciones predeterminado**. Esto significa que puede usar funciones XPath 2.0 y XSLT 2.0 en su hoja de estilos sin prefijos. Si declara el espacio de nombres Funciones XPath 2.0 en su hoja de estilos con un prefijo, podrá usar el prefijo asignado en la declaración.

- Cuando se usan constructores de tipo y tipos del espacio de nombres XML Schema, el prefijo utilizado en la declaración de espacio de nombres se debe usar en la llamada al constructor de tipo (por ejemplo, `xs:date`).
- Algunas funciones XPath 2.0 se llaman igual que algunos tipos de datos de XML Schema. Por ejemplo, las funciones XPath `fn:string` y `fn:boolean` y los tipos de datos de XML Schema `xs:string` y `xs:boolean`. Por tanto, si usa la expresión `string('Hello')`, la expresión se evalúa como `fn:string('Hello')` y no como `xs:string('Hello')`.

Compatibilidad con esquemas

El motor XSLT 2.0 está preparado para esquemas de modo que puede usar tipos de esquema definidos por el usuario y la instrucción `xsl:validate`.

Comportamiento propio de esta implementación

Más abajo puede ver cómo se ocupa el motor XSLT 2.0 de algunos aspectos del comportamiento de las funciones XSLT 2.0 relacionadas con esta implementación.

`xsl:result-document`

También son compatibles estas codificaciones específicas de Altova: `x-base16tobinary` y `x-base64tobinary`.

`function-available`

Esta función mira si hay funciones del ámbito disponibles (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`.

`unparsed-text-available`

El atributo `href` acepta (i) rutas de acceso relativas para archivos que estén en la carpeta del URI base y (ii) rutas de acceso absolutas con o sin el protocolo `file://`. También son compatibles estas codificaciones específicas de Altova: `x-binarytobase16` y `x-binarytobase64`.

Nota: estos valores de codificación estaban implementados en el ya descatalogado AltovaXML pero ya no se utilizan (son obsoletos): `base16tobinary`, `base64tobinary`, `binarytobase16` y `binarytobase64`.

11.1.1.3 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 está preparado para esquemas.

Codificación

El motor XQuery 1.0 es compatible con las codificaciones de caracteres UTF-8 y UTF-16.

Espacios de nombres

Se predefinen estos URI de espacios de nombres y sus enlaces asociados.

Espacio de nombres	Prefijo	URI del espacio de nombres
Tipos XML Schema	`xs:`	`http://www.w3.org/2001/XMLSchema`
Schema instance	`xsi:`	`http://www.w3.org/2001/XMLSchema-instance`
Funciones integradas	`fn:`	`http://www.w3.org/2005/xpath-functions`
Funciones locales	`local:`	`http://www.w3.org/2005/xquery-local-functions`

Es importante tener en cuenta que:

- El motor XQuery 1.0 entiende que los prefijos de la tabla anterior están enlazados con los correspondientes espacios de nombres.
- Como el espacio de nombres de funciones integradas (ver tabla) es el espacio de nombres de funciones predeterminado de XQuery, no es necesario usar el prefijo `fn:` cuando se invocan funciones integradas (por ejemplo, `string("Hello")` llamará a la función `fn:string`). No obstante, el prefijo `fn:` se puede utilizar para llamar a una función integrada sin necesidad de declarar el espacio de nombres en el prólogo de la consulta (por ejemplo: `fn:string("Hello")`).
- Puede cambiar el espacio de nombres de funciones predeterminado declarando la expresión `default function namespace` en el prólogo de la consulta.
- Cuando use tipos del espacio de nombres XML Schema, puede usar el prefijo `xs:` sin necesidad de declarar los espacios de nombres de forma explícita ni enlazar estos prefijos a los espacios de nombres en el prólogo de la consulta. (Ejemplo: `xs:date` y `xs:yearMonthDuration`.) Si quiere usar otros prefijos para el espacio de nombres de XML Schema, estos se deben declarar en el prólogo de la consulta. (Ejemplo: `declare namespace alt = "http://www.w3.org/2001/XMLSchema"; alt:date("2004-10-04")`.)
- Recuerde que los tipos de datos `untypedAtomic`, `dayTimeDuration` y `yearMonthDuration` se movieron del espacio de nombres XPath Datatypes al espacio de nombres XML Schema (es decir, ahora es `xs:yearMonthDuration`.)

Si se asignaron mal los espacios de nombres para funciones, constructores de tipo, pruebas de nodo, etc., se emite un error. Sin embargo, recuerde que algunas funciones se llaman igual que los tipos de datos de esquema (p. ej. `fn:string` y `fn:boolean`.) (Se definen `xs:string` y `xs:boolean`.) El prefijo del espacio de nombres determina si se usa la función o el constructor de tipo.

Documento XML de origen y validación

Los documentos XML que se utilizan para ejecutar un documento XQuery con el motor XQuery 1.0 deben tener un formato XML correcto. Sin embargo, no es necesario que sean válidos con respecto a un esquema XML. Si el archivo no es válido, el archivo no válido se carga sin información de esquema. Si el archivo XML está asociado a un esquema externo y es válido con respecto a dicho esquema, se genera información posterior a la validación de esquema, que se utilizará para evaluar la consulta.

Comprobación de tipos estática y dinámica

En la fase de análisis estático se revisan aspectos de la consulta como la sintaxis, si existen referencias externas (p. ej. para módulos), si las funciones y variables que se invocan están definidas, etc. Si se detecta un error en la fase de análisis estático, se notifica y la ejecución se interrumpe.

La comprobación dinámica de tipos se realiza en tiempo de ejecución, cuando la consulta se ejecuta. Si un tipo no es compatible con los requisitos de una operación, se emite un error. Por ejemplo, la expresión `xs:string("1") + 1` devuelve un error porque la operación de suma no se puede llevar a cabo en un operando de tipo `xs:string`.

Módulos biblioteca

Los módulos biblioteca almacenan funciones y variables para poder volver a utilizarlas. El motor XQuery 1.0 es compatible con el uso de módulos almacenados en un **solo archivo XQuery externo**. Dicho archivo de módulo debe incluir una declaración `module` en su prólogo que apunte a un espacio de nombres de destino Por ejemplo:

```
module namespace libns="urn:module-library";
declare variable $libns:company := "Altova";
declare function libns:webaddress() { "http://www.altova.com" };
```

Todas las funciones y variables declaradas en el módulo pertenecen al espacio de nombres asociado al módulo. El módulo se importa en un archivo XQuery con la instrucción `import module` del prólogo de la consulta. La instrucción `import module` solamente importa funciones y variables declaradas directamente en el archivo de módulo biblioteca. Por ejemplo:

```
import module namespace modlib = "urn:module-library" at "modulefilename.xq";

if        ($modlib:company = "Altova")
then       modlib:webaddress()
else       error("No match found.")
```

Funciones externas

Las funciones externas son incompatibles con el motor XQuery 1.0, es decir, todas las expresiones que usen la palabra clave `external`. Por ejemplo:

```
declare function hoo($param as xs:integer) as xs:string external;
```

Intercalaciones

La intercalación predeterminada es la intercalación de puntos de código Unicode, que compara las cadenas de texto según sus puntos de código Unicode. Otras intercalaciones compatibles son las intercalaciones ICU que se enumeran aquí. Para usar una intercalación concreta, indique su URI tal y como aparece en la lista de intercalaciones compatibles. Las comparaciones de cadenas de texto, incluidas las comparaciones para las funciones `fn:max` y `fn:min`, se harán según la intercalación especificada. Si no se indica la opción de intercalación, se utiliza la intercalación de puntos de código Unicode predeterminada.

Precisión de tipos numéricos

- El tipo de datos `xs:integer` es de precisión arbitraria, es decir, puede representar un número de dígitos cualquiera.
- El tipo de datos `xs:decimal` tiene un límite de 20 dígitos después del punto decimal.
- Los tipos de datos `xs:float` y `xs:double` tienen una precisión limitada de 15 dígitos.

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.

11.1.2 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.

Aspectos generales

Es necesario tener en cuenta estos puntos generales:

- A las funciones de las bibliotecas de funciones `core` definidas en las especificaciones W3C se les puede llamar sin un prefijo. Esto se debe a que los motores XSLT y XQuery leen funciones sin prefijo como si pertenecieran a un espacio de nombres de funciones predeterminado, que es el que se especifica en las especificaciones de las funciones XPath y XQuery`http://www.w3.org/2005/xpath-functions`. Si este espacio de nombres se declara explícitamente en un documento XSLT o XQuery, el prefijo utilizado en la declaración de espacio de nombres también se puede usar en el nombre de las funciones.
- Por lo general, si una función espera como argumento una secuencia de un elemento y se suministra una secuencia de más de un elemento, entonces se devuelve un error.
- Todas las comparaciones de cadena se realizan usando la intercalación de puntos de código Unicode.
- Los resultados que son QName se serializan de esta forma `[prefijo:]nombrelocal`.

Precisión de xs:decimal

El término *precisión* hace referencia al número de dígitos del número y la especificación exige un mínimo de 18 dígitos. Para las operaciones de división que dan un resultado de tipo `xs:decimal`, la precisión es de 19 dígitos después del punto decimal sin redondear.

Uso horario implícito

Cuando hay que comparar dos valores `date`, `time` o `dateTime`, es necesario conocer el uso horario de los valores que se deben comparar. Cuando el uso horario no se conoce de forma explícita, se usa el uso horario implícito, que se toma del reloj del sistema. Para probar cuál es su valor puede utilizar la función `fn:implicit-timezone()`.

Intercalaciones

La intercalación predeterminada es la intercalación de puntos de código Unicode, que compara las cadenas de texto según sus puntos de código Unicode. El motor usa el algoritmo de intercalación 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 `fn:max` y `fn:min`) se harán según la intercalación especificada. Si no se especifica la opción de intercalación, se usa la intercalación predeterminada de puntos de código Unicode.

Idioma	Identificadores URI
da: danés	da_DK
de: alemán	de_AT, de_BE, de_CH, de_DE, de_LI, de_LU
en: inglés	en_AS, en_AU, en_BB, en_BE, en_BM, en_BW, en_BZ, en_CA, en_GB, en_GU, en_HK, en_IE, en_IN, en_JM, en_MH, en_MP, en_MT, en_MU, en_NA, en_NZ, en_PH, en_PK, en_SG, en_TT, en_UM, en_US, en_VI, en_ZA, en_ZW
es: español	es_419, es_AR, es_BO, es_CL, es_CO, es_CR, es_DO, es_EC, es_ES, es_GQ, es_GT, es_HN, es_MX, es_NI, es_PA, es_PE, es_PR, es_PY, es_SV, es_US, es_UY, es_VE
fr: francés	fr_BE, fr_BF, fr_BI, fr_BJ, fr_BL, fr_CA, fr_CD, fr_CF, fr_CG, fr_CH, fr_CI, fr_CM, fr_DJ, fr_FR, fr_GA, fr_GN, fr_GP, fr_GQ, fr_KM, fr_LU, fr_MC, fr_MF, fr_MG, fr_ML, fr_MQ, fr_NE, fr_RE, fr_RW, fr_SN, fr_TD, fr_TG
it: italiano	it_CH, it_IT
ja: jaoonés	ja_JP
nb: noruego bokmål	nb_NO
nl: holandés	nl_AW, nl_BE, nl_NL
nn: noruego nynorsk	nn_NO
pt: portugués	pt_AO, pt_BR, pt_GW, pt_MZ, pt_PT, pt_ST
ru: ruso	ru_MD, ru_RU, ru_UA
sv: sueco	sv_FI, sv_SE

Eje del espacio de nombres
El eje del espacio de nombres se dejó de utilizar en XPath 2.0. Sin embargo, los ejes de espacio de nombres son compatibles con la aplicación. Para acceder a la información sobre el espacio de nombres con mecanismos de XPath 2.0, utilice las funciones `fn:in-scope-prefixes()`, `fn:namespace-uri()` y `fn:namespace-uri-for-prefix()`.

11.1.2.1 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**
Funciones XSLT *(en expresiones XPath en XSLT):*	**XSLT1** **XSLT2** **XSLT3**
Funciones XQuery *(en expresiones XQuery en XQuery):*	**XQ1** **XQ3.1**

Funciones XSLT

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:

- Funciones de fecha y hora
- Funciones de geoubicación
- Funciones relacionadas con imágenes
- Funciones numéricas
- Funciones de secuencia
- Funciones de cadena
- Funciones varias

11.1.2.1.1 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
Funciones XSLT (en expresiones XPath en XSLT):	XSLT1 XSLT2 XSLT3
Funciones XQuery (en expresiones XQuery en XQuery):	XQ1 XQ3.1

Funciones estándar

▼ 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]

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]

11.1.2.1.2 Funciones XPath/XQuery: de fecha y hora

Las funciones de extensión de fecha y hora de Altova se pueden usar en expresiones XPath y XQuery y permiten procesar datos almacenados en tipos de datos XML Schema de fecha y hora. Estas funciones se pueden usar con los **motores XPath 3.0 y XQuery 3.0** de Altova y están disponibles en contextos XPath/XQuery.

Nota sobre el nombre de las funciones y lenguajes

Puede utilizar todas las funciones de extensión de Altova en sus expresiones XPath/XQuery. Con ellas conseguirá funciones adicionales no disponibles en la biblioteca de funciones estándar de XPath, XQuery y XSLT. Las funciones de extensión de Altova están en el **espacio de nombres** `http://www.altova.com/xslt-extensions` y en esta sección se presentan con el prefijo, que se supone estará enlazado al espacio de nombres señalado. Tenga en cuenta que en futuras versiones del producto algunas funciones pueden dejar de ser compatibles o su comportamiento puede cambiar. Por tanto, consulte siempre la documentación del producto para conocer el funcionamiento de estas funciones en cada versión del producto.

Funciones XPath (en expresiones XPath en XSLT):	XP1 XP2 XP3.1
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

- Agregar una duración a xs:dateTime y devolver xs:dateTime
- Agregar una duración a xs:date y devolver xs:date
- Agregar una duración a xs:time y devolver xs:time
- Recuperar duraciones y aplicarles formato
- Quitar la zona horaria de las funciones que generan la fecha/hora actual
- Devolver el día de la semana de una fecha como número entero
- Devolver el número de semana de una fecha como número entero
- Generar la fecha, la hora y el tipo de duración a partir de los componentes léxicos de cada tipo
- Construir un tipo date, dateTime o a partir de la cadena de entrada
- Funciones para calcular la edad

▼ Funciones por orden alfabético

`altova:add-days-to-date`

[Subir]

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`+01:00 (por ejemplo)`.

▼ add-years-to-dateTime [altova:]

`altova:add-years-to-dateTime(FechaHora` *`as xs:dateTime`*`, Años` *`as xs:integer`*`)`
como **`xs:dateTime`** XP3.1 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]

Recuperar duraciones y aplicarles formato XP3.1 XQ3.1

Estas funciones añaden una duración a **xs:date** y devuelven **xs:date**. El tipo `xs:date` tiene el formato `SSAA-MM-DD`.

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]

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]

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]

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-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-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-time-no-TZ [altova:]

altova:current-time-no-TZ() as **xs:time** XP3.1 XQ3.1

Esta función no toma ningún argumento. Quita la parte de zona horaria de current-time() (que es la hora actual según el reloj del sistema) y devuelve un valor de tipo xs:time.

▣ *Ejemplos*

Si la hora actual es **14:00:00+01:00**:

- **altova:current-time-no-TZ**() devuelve 14:00:00

[Subir]

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(FechaHora *as xs:dateTime***)** como **xs:integer** XP3.1 XQ3.1

Toma una fecha con hora como único argumento y devuelve el día de la semana de la fecha dada como número entero. Los días de la semana se numeran del 1 al 7 empezando por Domingo=1. Si necesita usar el formato europeo (donde Lunes=1), utilice la otra firma de esta función (*ver más abajo*).

Ejemplos

- **altova:weekday-from-dateTime**(xs:dateTime("2014-02-03T09:00:00")) devuelve 2, lo cual significa "Lunes".

altova:weekday-from-dateTime(DateTime *as xs:dateTime*, **Formato** *as xs:integer***)** como **xs:integer** XP3.1 XQ3.1

Toma una fecha con hora como primer argumento y devuelve el día de la semana de la fecha dada como número entero. Los días de la semana se numeran del 1 al 7 empezando por Lunes=1. Si el segundo argumento (**Formato**) es 0, entonces los días de la semana se numeran del 1 al 7 empezando por Domingo=1. Si el segundo argumento es un entero distinto de 0, entonces Lunes=1. Si falta el segundo argumento, la función se lee como en la firma anterior (*ver más arriba*).

Ejemplos

- **altova:weekday-from-dateTime**(xs:dateTime("2014-02-03T09:00:00"), 1) devuelve 1, lo cual significa "Lunes".
- **altova:weekday-from-dateTime**(xs:dateTime("2014-02-03T09:00:00"), 4) devuelve 1, lo cual significa "Lunes".
- **altova:weekday-from-dateTime**(xs:dateTime("2014-02-03T09:00:00"), 0) devuelve 2, lo cual significa "Lunes".

▼ weekday-from-date [altova:]

altova:weekday-from-date(Fecha *as xs:date***)** como **xs:integer** XP3.1 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(Fecha` *`as xs:date`*`, `**`Formato`** *`as 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. Los días de la semana se numeran del 1 al 7 empezando por `Lunes=1`. Si el segundo argumento (**`Formato`**) es `0`, entonces los días de la semana se numeran del 1 al 7 empezando por `Domingo=1`. Si el segundo argumento es un entero distinto de `0`, entonces `Lunes=1`. Si falta el segundo argumento, la función se lee como en la firma anterior (*ver más arriba*).

Ejemplos

- **`altova:weekday-from-date`**`(xs:date("2014-02-03"), 1)` devuelve `1`, lo cual significa "Lunes"
- **`altova:weekday-from-date`**`(xs:date("2014-02-03"), 4)` devuelve `1`, lo cual significa "Lunes"
- **`altova:weekday-from-date`**`(xs:date("2014-02-03"), 0)` devuelve `2`, lo cual significa "Lunes"

[Subir]

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]

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

▼ 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]

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]

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]

11.1.2.1.3 Funciones XPath/XQuery: de 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
Funciones XSLT (en expresiones XPath en XSLT):	XSLT1 XSLT2 XSLT3
Funciones XQuery (en expresiones XQuery en XQuery):	XQ1 XQ3.1

▼ 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` y el atributo `@Geolocation` de los metadatos Exif se pueden utilizar para suministrar la cadena de entrada de la geoubicación (*ver ejemplos*).

⊟ *Ejemplos*

- `altova:parse-geolocation("33.33 -22.22")` devuelve la secuencia de dos `xs:decimals (33.33, 22.22)`
- `altova:parse-geolocation("48°51'29.6""N 24°17'40.2""")` devuelve la secuencia de dos `xs:decimals (48.8582222222222, 24.2945)`
- `altova:parse-geolocation('48°51''29.6"N 24°17''40.2"')` devuelve la secuencia de dos `xs:decimals (48.8582222222222, 24.2945)`
- `altova:parse-geolocation( image-exif-data(//MisImágenes/Imagen20141130.01)/@Geolocation )` devuelve una secuencia de dos `xs:decimals`

⊟ *Formato de las cadenas de entrada de geoubicaciones:*

La cadena de entrada de la geoubicación debe contener la latitud y la longitud (en ese orden) se paradas por un espacio en blanco. Ambas pueden estar en cualquier formato de los que se indican más abajo y puede combinar formatos distintos. Es decir, la latitud puede estar en un formato y la longitud en otro. Los valores de la latitud deben estar comprendidos entre `+90` y `-90` (`N` a `S`). Los valores de longitud deben estar comprendidos entre `+180` y `-180` (`E` a `W`).

Nota: si utiliza comillas simples o dobles para delimitar el argumento de la cadena de entrada, esto dará lugar a un conflicto con las comillas simples o dobles que se utilizan, respectivamente, para indicar los valores de los minutos y los segundos. Si esto ocurre, debe añadir caracteres de escape a las comillas utilizadas para los minutos y segundos (esto se hace duplicando las comillas.

- Grados, minutos y segundos decimales + orientación como sufijo (`N/S`, `W/E`)
  `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/W`**) 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`, `W/E`)
  `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/W**) es opcional
  +/-D°M.MM' +/-D°M.MM'
  Ejemplo: **+33°55.55' -22°44.44'**

- Grados decimales + orientación como sufijo (N/S, W/E)
  D.DDN/S D.DDW/E
  Ejemplo: **33.33N 22.22W**

- Grados decimales + prefijo (+/-). El signo + para (**N/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 y el atributo de metadatos Exif @Geolocation pueden utilizarse para suministrar las cadenas de entrada de geoubicaciones.

Ejemplos

- **altova:geolocation-distance-km**("33.33 -22.22", "48°51'29.6""N 24° 17'40.2""") devuelve el xs:decimal 4183.08132372392

Formato de las cadenas de entrada de geoubicaciones:

La cadena de entrada de la geoubicación debe contener la latitud y la longitud (en ese orden) se paradas por un espacio en blanco. Ambas pueden estar en cualquier formato

de los que se indican más abajo y puede combinar formatos distintos. Es decir, la latitud puede estar en un formato y la longitud en otro. Los valores de la latitud deben estar comprendidos entre `+90` y `-90` (`N` a `S`). Los valores de longitud deben estar comprendidos entre `+180` y `-180` (`E` a `W`).

Nota: si utiliza comillas simples o dobles para delimitar el argumento de la cadena de entrada, esto dará lugar a un conflicto con las comillas simples o dobles que se utilizan, respectivamente, para indicar los valores de los minutos y los segundos. Si esto ocurre, debe añadir caracteres de escape a las comillas utilizadas para los minutos y segundos (esto se hace duplicando las comillas.

- Grados, minutos y segundos decimales + orientación como sufijo (`N/S`, `W/E`)
  `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/W`**) 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`, `W/E`)
  `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/W`**) es opcional
  `+/-D°M.MM'  +/-D°M.MM'`
  Ejemplo: **`+33°55.55'  -22°44.44'`**

- Grados decimales + orientación como sufijo (`N/S`, `W/E`)
  `D.DDN/S  D.DDW/E`
  Ejemplo: **`33.33N  22.22W`**

- Grados decimales + prefijo (`+/-`). El signo + para (**`N/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	S	151 13	E	33°51'21.91"S

21.91		11.73		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` y el atributo de metadatos Exif `@Geolocation` pueden utilizarse para suministrar las cadenas de entrada de geoubicaciones.

⊟ *Ejemplos*

- **altova:geolocation-distance-mi**(`"33.33 -22.22"`, `"48°51'29.6""N 24° 17'40.2"""`) devuelve el `xs:decimal` `2599.40652340653`

⊟ *Formato de las cadenas de entrada de geoubicaciones:*

La cadena de entrada de la geoubicación debe contener la latitud y la longitud (en ese orden) se paradas por un espacio en blanco. Ambas pueden estar en cualquier formato de los que se indican más abajo y puede combinar formatos distintos. Es decir, la latitud puede estar en un formato y la longitud en otro. Los valores de la latitud deben estar comprendidos entre `+90` y `-90` (`N` a `S`). Los valores de longitud deben estar comprendidos entre `+180` y `-180` (`E` a `W`).

Nota: si utiliza comillas simples o dobles para delimitar el argumento de la cadena de entrada, esto dará lugar a un conflicto con las comillas simples o dobles que se utilizan, respectivamente, para indicar los valores de los minutos y los segundos. Si esto ocurre, debe añadir caracteres de escape a las comillas utilizadas para los minutos y segundos (esto se hace duplicando las comillas.

- Grados, minutos y segundos decimales + orientación como sufijo (`N/S`, `W/E`)
  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/W**) 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`, `W/E`)
  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/W**) es opcional
  +/-D°M.MM' +/-D°M.MM'

Ejemplo: **+33°55.55' -22°44.44'**

- Grados decimales + orientación como sufijo (`N/S`, `W/E`)
  D.DDN/S D.DDW/E
  Ejemplo: **33.33N 22.22W**

- Grados decimales + prefijo (`+/-`). El signo + para (**N/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-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` y el atributo de metadatos Exif `@Geolocation` 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.

  - Grados, minutos y segundos decimales + orientación como sufijo (`N/S`, `W/E`)
    `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/W`**) 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`, `W/E`)
    `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/W`**) es opcional
    `+/-D°M.MM'  +/-D°M.MM'`
    Ejemplo: **`+33°55.55'  -22°44.44'`**

  - Grados decimales + orientación como sufijo (`N/S`, `W/E`)
    `D.DDN/S  D.DDW/E`
    Ejemplo: **`33.33N  22.22W`**

  - Grados decimales + prefijo (`+/-`). El signo + para (**`N/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` y el atributo de metadatos Exif `@Geolocation` pueden utilizarse para suministrar las cadenas de entrada de geoubicaciones.

▫ *Ejemplos*

- **altova:geolocation-within-rectangle(**"33 -22", "58 -32", "-48 24") devuelve `true()`
- **altova:geolocation-within-rectangle(**"33 -22", "58 -32", "48 24") devuelve `false()`
- **altova:geolocation-within-rectangle(**"33 -22", "58 -32", "48°51'29.6""S 24°17'40.2""") devuelve `true()`

▫ *Formato de las cadenas de entrada de geoubicaciones:*

La cadena de entrada de la geoubicación debe contener la latitud y la longitud (en ese orden) se paradas por un espacio en blanco. Ambas pueden estar en cualquier formato de los que se indican más abajo y puede combinar formatos distintos. Es decir, la latitud puede estar en un formato y la longitud en otro. Los valores de la latitud deben estar comprendidos entre `+90` y `-90` (`N` a `S`). Los valores de longitud deben estar comprendidos entre `+180` y `-180` (`E` a `W`).

Nota: si utiliza comillas simples o dobles para delimitar el argumento de la cadena de entrada, esto dará lugar a un conflicto con las comillas simples o dobles que se utilizan, respectivamente, para indicar los valores de los minutos y los segundos. Si esto ocurre, debe añadir caracteres de escape a las comillas utilizadas para los minutos y segundos (esto se hace duplicando las comillas.

- Grados, minutos y segundos decimales + orientación como sufijo (N/S, W/E)
  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/W**) 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, W/E)
  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/W**) es opcional
  +/-D°M.MM' +/-D°M.MM'
  Ejemplo: **+33°55.55' -22°44.44'**

- Grados decimales + orientación como sufijo (N/S, W/E)
  D.DDN/S D.DDW/E
  Ejemplo: **33.33N 22.22W**

- Grados decimales + prefijo (+/-). El signo + para (**N/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]

11.1.2.1.4 Funciones XPath/XQuery: relacionadas con imágenes

Las funciones de extensión XPath/XQuery para trabajar con imágenes son compatibles con la versión actual de 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
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:

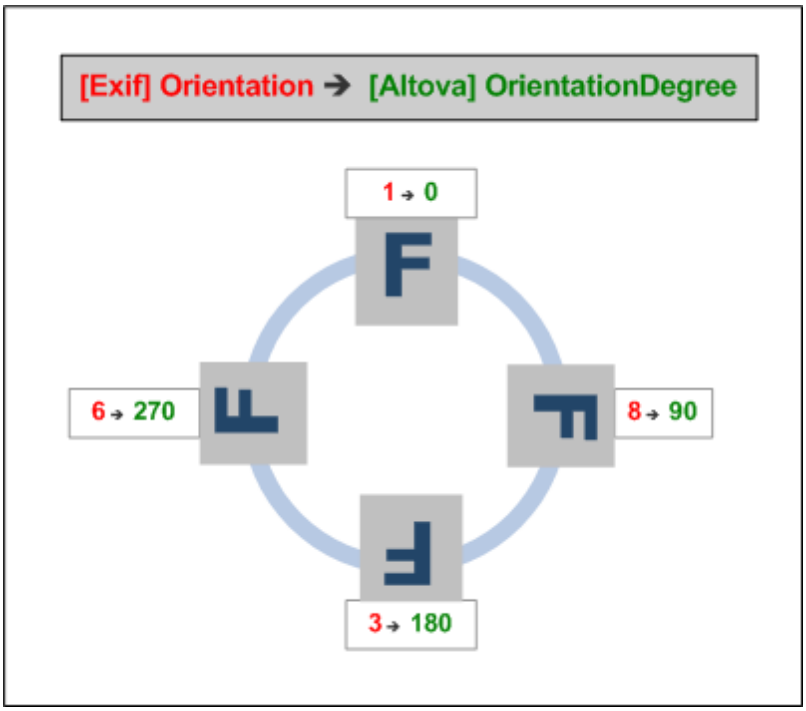
GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151°13'11.73"E

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]

11.1.2.1.5 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
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]

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]

Funciones de formato numérico

▼ 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")`.

[Subir]

11.1.2.1.6 Funciones XPath/XQuery: de secuencia

Las funciones de extensión de Altova para trabajar con secuencias pueden utilizarse en expresiones XPath y XQuery y ofrecen funciones adicionales para el procesamiento de datos. Estas funciones se pueden usar con los motores **XPath 3.0** y **XQuery 3.0** de Altova. Están disponibles en contextos XPath/XQuery.

Nota sobre el nombre de las funciones y lenguajes

Puede utilizar todas las funciones de extensión de Altova en sus expresiones XPath/XQuery. Con ellas conseguirá funciones adicionales no disponibles en la biblioteca de funciones estándar de XPath, XQuery y XSLT. Las funciones de extensión de Altova están en el **espacio de nombres** `http://www.altova.com/xslt-extensions` y en esta sección se presentan con el prefijo, que se supone estará enlazado al espacio de nombres señalado. Tenga en cuenta que en futuras versiones del producto algunas funciones pueden dejar de ser compatibles o su

comportamiento puede cambiar. Por tanto, consulte siempre la documentación del producto para conocer el funcionamiento de estas funciones en cada versión del producto.

Funciones XPath (en expresiones XPath en XSLT):	XP1 XP2 XP3.1
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 *como item()**, **(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.

▼ 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)

11.1.2.1.7 Funciones XPath/XQuery: de cadena

Las funciones de extensión de Altova para trabajar con cadenas pueden utilizarse en expresiones XPath y XQuery y ofrecen funciones adicionales para el procesamiento de datos. Estas funciones se pueden usar con los motores **XPath 3.0** y **XQuery 3.0** de Altova. Están disponibles en contextos XPath/XQuery.

Nota sobre el nombre de las funciones y lenguajes

Puede utilizar todas las funciones de extensión de Altova en sus expresiones XPath/XQuery. Con ellas conseguirá funciones adicionales no disponibles en la biblioteca de funciones estándar de XPath, XQuery y XSLT. Las funciones de extensión de Altova están en el **espacio de nombres** `http://www.altova.com/xslt-extensions` y en esta sección se presentan con el prefijo, que se supone estará enlazado al espacio de nombres señalado. Tenga en cuenta que en futuras versiones del producto algunas funciones pueden dejar de ser compatibles o su comportamiento puede cambiar. Por tanto, consulte siempre la documentación del producto para conocer el funcionamiento de estas funciones en cada versión del producto.

Funciones XPath (en expresiones XPath en XSLT):	XP1 XP2 XP3.1
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

▼ 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

▼ 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

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"

11.1.2.1.8 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
Funciones XSLT (en expresiones XPath en XSLT):	XSLT1 XSLT2 XSLT3
Funciones XQuery (en expresiones XQuery en XQuery):	XQ1 XQ3.1

▼ get-temp-folder [altova:]

altova:get-temp-folder() como xs:string XP2 XQ1 XP3.1 XQ3.1
Esta función no toma ningún argumento. Devuelve la ruta de acceso de la carpeta temporal del usuario actual.

⊟ *Ejemplo*

- **altova:get-temp-folder**()
  en un equipo Windows devuelve (más o menos) C:\Usuarios\<Usuario>\AppData\Local\Temp\ como valor de tipo xs:string.

▼ generate-guid [altova:]

altova:generate-guid() as **xs:string** XP2 XQ1 XP3.1 XQ3.1
Genera una cadena única de la interfaz gráfica del usuario.
⊞ *Ejemplo*

- **altova:generate-guid**() devuelve (por ejemplo) 85F971DA-17F3-4E4E-994E-

```
99137873ACCD
```

[Subir]

11.1.2.2 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 y .NET, así como scripts MSXSL para XSLT.

Esta sección describe cómo usar funciones de extensión y scripts MSXSL en hojas de estilos XSLT y documentos XQuery. Las funciones de extensión pueden organizarse en varios grupos:

- Funciones de extensión Java
- Funciones de extensión .NET
- Scripts MSXSL para XSLT

En los apartados de esta sección nos ocupamos de tres aspectos fundamentales: (i) cómo se llaman las funciones en sus respectivas bibliotecas, (ii) qué reglas deben seguirse para convertir los argumentos de una llamada a función en el formato de entrada necesario de la función y (iii) qué reglas deben seguirse para la conversión del tipo devuelto.

Requisitos
Para que estas funciones de extensión funcionen es necesario tener Java Runtime Environment (para las funciones Java) y .NET Framework 2.0 o superior (para las funciones .NET) instalado en el equipo que ejecuta la transformación XSLT o XQuery.

11.1.2.2.1 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:

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 y Archivos Jar definidos por el usuario. Recuerde que debe especificar las rutas de acceso de los archivos de clases que no están en el directorio actual y de todos los archivos JAR.

11.1.2.2.1.1 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*.)
- El archivo de clases no está en un paquete. El archivo XSLT/XQuery está en la misma carpeta que el archivo de clases. (*ver ejemplo*.)
- El archivo de clases está en un paquete. El archivo XSLT/XQuery está en una carpeta cualquiera. (*ver ejemplo*.)
- El archivo de clases no está en un paquete. El archivo XSLT/XQuery está una carpeta cualquiera. (*ver ejemplo*.)

Imaginemos que tenemos un archivo de clases que no está en un paquete y que está en la misma carpeta que el documento XSLT/XQuery. En este caso, puesto que en la carpeta se encuentran todas las clases, no es necesario especificar la ubicación del archivo. La sintaxis que se utiliza para identificar una clase es esta:

```
java:nombreClase
```

donde

`java:` indica que se está llamando a una función definida por el usuario (por defecto se cargan las clases Java del directorio actual)

`nombreClase` es el nombre de la clase del método elegido

La clase se identifica en un URI de espacio de nombres y el espacio de nombres se usa como prefijo para la llamada al método.

El archivo de clases está en un paquete. El archivo XSLT/XQuery está en la misma carpeta que el paquete Java

El código que aparece a continuación llama al método `getVehicleType()` de la clase `Car` del

paquete com.altova.extfunc. El paquete com.altova.extfunc está en la carpeta JavaProject. El archivo XSLT también está en la carpeta JavaProject.

```
<xsl:stylesheet version="2.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      xmlns:xs="http://www.w3.org/2001/XMLSchema"
      xmlns:fn="http://www.w3.org/2005/xpath-functions"
      xmlns:car="java:com.altova.extfunc.Car" >
<xsl:output exclude-result-prefixes="fn car xsl fo xs"/>

<xsl:template match="/">
   <a>
   <xsl:value-of select="car:getVehicleType()"/>
   </a>
</xsl:template>

</xsl:stylesheet>
```

El archivo de clases no está en un paquete. El archivo XSLT/XQuery está en la misma carpeta que el archivo de clases

El código que aparece a continuación llama al método getVehicleType() de la clase Car del paquete com.altova.extfunc. El archivo de clases Car está en esta carpeta: JavaProject/com/altova/extfunc. El archivo XSLT también está en la carpeta JavaProject/com/altova/extfunc.

```
<xsl:stylesheet version="2.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      xmlns:xs="http://www.w3.org/2001/XMLSchema"
      xmlns:fn="http://www.w3.org/2005/xpath-functions"
      xmlns:car="java:Car" >
<xsl:output exclude-result-prefixes="fn car xsl fo xs"/>

<xsl:template match="/">
   <a>
   <xsl:value-of select="car:getVehicleType()"/>
   </a>
</xsl:template>

</xsl:stylesheet>
```

El archivo de clases está en un paquete. El archivo XSLT/XQuery está en una carpeta cualquiera

El código que aparece a continuación llama al método getCarColor() de la clase Car del paquete com.altova.extfunc. El paquete com.altova.extfunc está en la carpeta JavaProject. El archivo XSLT está en otra carpeta cualquiera. En este caso debe especificarse la ubicación del paquete dentro del URI como una cadena de consulta. La sintaxis es esta:

```
java:nombreClase[?ruta=uri-del-paquete]
```

donde

`java`: indica que se está llamando a una función Java definida por el usuario
`uri-del-paquete` es el URI del paquete Java
`nombreClase` es el nombre de la clase del método elegido

La clase se identifica en un URI de espacio de nombres y el espacio de nombres se usa como prefijo para la llamada al método. El ejemplo de código que aparece a continuación explica cómo se accede a un archivo de clases que está ubicado en un directorio que no es el directorio actual.

```
<xsl:stylesheet version="2.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      xmlns:xs="http://www.w3.org/2001/XMLSchema"
      xmlns:fn="http://www.w3.org/2005/xpath-functions"
      xmlns:car="java:com.altova.extfunc.Car?path=file:///C:/
JavaProject/" >

<xsl:output exclude-result-prefixes="fn car xsl xs"/>

<xsl:template match="/">
   <xsl:variable name="myCar" select="car:new('red')" />
   <a><xsl:value-of select="car:getCarColor($myCar)"/></a>
</xsl:template>

</xsl:stylesheet>
```

El archivo de clases no está en un paquete. El archivo XSLT/XQuery está una carpeta cualquiera

El código que aparece a continuación llama al método `getCarColor()`de la clase `Car` del paquete `com.altova.extfunc`. El paquete `com.altova.extfunc` está en la carpeta `JavaProject`. El archivo XSLT está en otra carpeta cualquiera. En este caso debe especificarse la ubicación del paquete dentro del URI como una cadena de consulta. La sintaxis es esta:

```
java:nombreClase[?ruta=uri-del-archivoClases]
```

donde

`java`: indica que se está llamando a una función Java definida por el usuario
`uri-del-archivoClases` es el URI de la carpeta donde se ubica el archivo de clases
`nombreClase` es el nombre de la clase del método elegido

La clase se identifica en un URI de espacio de nombres y el espacio de nombres se usa como prefijo para la llamada al método. El ejemplo de código que aparece a continuación explica cómo se accede a un archivo de clases que está ubicado en un directorio que no es el directorio actual.

```
<xsl:stylesheet version="2.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      xmlns:xs="http://www.w3.org/2001/XMLSchema"
      xmlns:fn="http://www.w3.org/2005/xpath-functions"
      xmlns:car="java:Car?path=file:///C:/JavaProject/com/altova/
extfunc/" >

<xsl:output exclude-result-prefixes="fn car xsl xs"/>

<xsl:template match="/">
   <xsl:variable name="myCar" select="car:new('red')" />
   <a><xsl:value-of select="car:getCarColor($myCar)"/></a>
</xsl:template>

</xsl:stylesheet>
```

Nota: cuando se presenta una ruta de acceso por medio de una función de extensión, la ruta de acceso se añade al ClassLoader.

11.1.2.2.1.2 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.

11.1.2.2.1.3 Constructores

Una función de extensión se puede usar para llamar a un constructor Java. A todos los constructores se les llama con la pseudofunción `new()`.

Si el resultado de una llamada a un constructor Java se puede convertir de manera implícita a tipos de datos XPath/XQuery, entonces la llamada a la función de extensión Java devuelve una secuencia que es un tipo de datos XPath/XQuery. Si el resultado de una llamada a un constructor Java no se puede convertir a un tipo de datos XPath/XQuery adecuado, entonces el constructor crea un objeto Java contenido con un tipo que es el nombre de la clase que devuelve ese objeto Java. Por ejemplo, si se llama a un constructor para la clase `java.util.Date` (`java.util.Date.new()`), entonces se devuelve un objeto que tiene el tipo `java.util.Date`. Puede que el formato léxico del objeto devuelto no coincida con el formato léxico de un tipo de datos XPath y, por tanto, su valor debe convertirse al formato léxico del tipo de datos XPath pertinente y después al tipo de datos XPath.

Puede hacer dos cosas con el objeto Java creado por un constructor:

- Puede asignar el objeto a una variable:
  ```
  <xsl:variable name="currentdate" select="date:new()"
  xmlns:date="java:java.util.Date" />
  ```
- Puede pasar el objeto a una función de extensión (*ver métodos de instancia y campos de instancia*):
  ```
  <xsl:value-of select="date:toString(date:new())"
  xmlns:date="java:java.util.Date" />
  ```


11.1.2.2.1.4 Métodos estáticos y campos estáticos

La llamada a un método estático la hace directamente su nombre Java y se hace presentando los argumentos para el método. A los campos estáticos (es decir, los métodos que no toman argumentos), como los campos de valor constante `E` y `PI`, se accede sin especificar ningún argumento.

Ejemplos de código XSLT

Aquí puede ver varios ejemplos de cómo se llama a métodos y campos estáticos:

```
<xsl:value-of xmlns:jMath="java:java.lang.Math"
              select="jMath:cos(3.14)" />

<xsl:value-of xmlns:jMath="java:java.lang.Math"
              select="jMath:cos( jMath:PI() )" />

<xsl:value-of xmlns:jMath="java:java.lang.Math"
              select="jMath:E() * jMath:cos(3.14)" />
```

Observe que las funciones de extensión anteriores tienen el formato `prefijo:nombreFunción()`. En los tres ejemplos anteriores, el prefijo es `jMath:`, que está asociado al URI de espacio de nombres `java:java.lang.Math`. (El URI de espacio de nombres debe empezar por `java:`. En los ejemplos anteriores se extiende para contener el nombre de la clase (`java.lang.Math`).) La parte `nombreFunción()` de las funciones de extensión debe coincidir con el nombre de una clase pública (p. ej. `java.lang.Math`) seguido del nombre de un método estático público con sus argumentos (como `cos(3.14)`) o de un campo estático público (como `PI()`).

En los tres ejemplos anteriores, el nombre de la clase se incluyó en el URI de espacio de nombres. Si no estuviera en el URI de espacio de nombres, se incluiría en la parte `nombreFunción()` de la función de extensión. Por ejemplo:

```
<xsl:value-of xmlns:java="java:"
              select="java:java.lang.Math.cos(3.14)" />
```

Ejemplo de XQuery

Un ejemplo de XQuery similar sería:

```
<cosine xmlns:jMath="java:java.lang.Math">
   {jMath:cos(3.14)}
</cosine>
```

11.1.2.2.1.5 Métodos de instancia y campos de instancia

A un método de instancia se le pasa un objeto Java como primer argumento de la llamada a método. Dicho objeto Java suele crearse usando una función de extensión (por ejemplo, una llamada a un constructor) o un parámetro o una variable de hoja de estilos. Un ejemplo de código XSLT de este tipo sería:

```
<xsl:stylesheet version="1.0" exclude-result-prefixes="date"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:date="java:java.util.Date"
  xmlns:jlang="java:java.lang">
  <xsl:param name="CurrentDate" select="date:new()"/>
  <xsl:template match="/">
      <enrollment institution-id="Altova School"
                date="{date:toString($CurrentDate)}"
                type="
{jlang:Object.toString(jlang:Object.getClass( date:new() ))}">
      </enrollment>
  </xsl:template>
</xsl:stylesheet>
```

En el ejemplo anterior el valor del nodo `enrollment/@type` se crea de la siguiente manera:

1. Se crea un objeto con un constructor para la clase `java.util.Date` (con el constructor `date:new()`).
2. Este objeto Java se pasa como argumento del método `jlang.Object.getClass`.
3. El objeto que obtiene el método `getClass` se pasa como argumento al método `jlang.Object.toString`.

El resultado (el valor de `@type`) será una cadena con este valor: `java.util.Date`.

En teoría, un campo de instancia es diferente de un método de instancia porque al campo de instancia no se pasa como argumento un objeto Java propiamente dicho. En su lugar se pasa como argumento un parámetro o variable. Sin embargo, el parámetro o la variable puede contener el valor devuelto por un objeto Java. Por ejemplo, el parámetro `CurrentDate` toma el valor que devolvió un constructor para la clase `java.util.Date`. Este valor se pasa después como argumento al método de instancia `date:toString` a fin de suministrar el valor de `/enrollment/@date`.

11.1.2.2.1.6 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.

11.1.2.2.1.7 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).

11.1.2.2.2 Funciones de extensión .NET

Si trabaja en la plataforma .NET desde un equipo Windows, puede usar funciones de extensión escritas en cualquier lenguaje .NET (p. ej. C#). Una función de extensión .NET se puede usar dentro de una expresión XPath/XQuery para invocar un constructor, una propiedad o un método (estático o de instancia) de una clase .NET.

A una propiedad de una clase .NET se le llama usando la sintaxis `get_NombrePropiedad()`.

Este apartado tiene varias partes:

- Constructores
- Métodos estáticos y campos estáticos
- Métodos de instancia y campos de instancia
- Tipos de datos: conversión de XPath/XQuery en .NET
- Tipos de datos: conversión de .NET en XPath/XQuery

Formato de la función de extensión

La función de extensión de la expresión XPath/XQuery debe tener este formato `prefijo:nombreFunción()`.

- La parte `prefijo:` está asociada a un URI que identifica la clase .NET.
- La parte `nombreFunción()` identifica el constructor, la propiedad o el método (estático o de instancia) dentro de la clase .NET y, si es necesario, suministra los argumentos.
- El URI debe empezar por `clitype:` (que identifica la función como función de extensión .NET).
- El formato `prefijo:nombreFunción()` de la función de extensión se puede usar con clases del sistema y con clases de un ensamblado cargado. No obstante, si se tiene que cargar una clase, será necesario suministrar parámetros que contengan la información necesaria.

Parámetros

Para cargar un ensamblado se usan estos parámetros:

`asm`	El nombre del ensamblado que se debe cargar.
`ver`	El número de versión (máximo cuatro enteros separados por puntos).
`sn`	El símbolo de clave del nombre seguro del ensamblado (16 dígitos hexadecimales).
`from`	Un URI que da la ubicación del ensamblado (DLL) que se debe cargar. Si el URI es relativo, es relativo al archivo XSLT o XQuery. Si está presente este parámetro, se ignoran los demás parámetros.
`partialname`	El nombre parcial del ensamblado. Se suministra a `Assembly.LoadWith.PartialName()`, que intentará cargar el ensamblado. Si está presente el parámetro `partialname`, se ignoran los demás parámetros.
`loc`	La configuración regional, por ejemplo, `en-US`. La configuración predeterminada es `neutral`.

Si el ensamblado se debe cargar desde un archivo DLL, use el parámetro `from` y omita el parámetro `sn`. Si el ensamblado se debe cargar desde el caché general de ensamblados (GAC), use el parámetro `sn` y omita el parámetro `from`.

Debe insertar un signo de interrogación final antes del primer parámetro y los parámetros deben separarse con un punto y coma (;). El nombre de parámetro da su valor con un signo igual (=), como en el ejemplo que aparece más abajo.

Ejemplos de declaraciones de espacios de nombres

Esto es un ejemplo de una declaración de espacio de nombres en XSLT que identifica la clase del sistema `System.Environment`:

```
xmlns:myns="clitype:System.Environment"
```

Esto es un ejemplo de una declaración de espacio de nombres en XSLT que identifica la clase que se debe cargar como `Trade.Forward.Scrip`:

```
xmlns:myns="clitype:Trade.Forward.Scrip?asm=forward;version=10.6.2.1"
```

Esto es un ejemplo de una declaración de espacio de nombres en XQuery que identifica la clase del sistema `MyManagedDLL.testClass`:. Existen dos tipos de clases:

1. Cuando el ensamblado se carga desde el GAC:
   ```
   declare namespace cs="clitype:MyManagedDLL.testClass?asm=MyManagedDLL;
          ver=1.2.3.4;loc=neutral;sn=b9f091b72dccfba8";
   ```

2. Cuando el ensamblado se carga desde el archivo DLL (ver las referencias parciales y completas):
   ```
          declare namespace cs="clitype:MyManagedDLL.testClass?from=file:///
   C:/Altova
          Projects/extFunctions/MyManagedDLL.dll;

      declare namespace cs="clitype:MyManagedDLL.testClass?
   ```

```
from=MyManagedDLL.dll;
```

Ejemplo de código XSLT

Aquí puede ver un ejemplo de código XSLT que llama a funciones de la clase del sistema `System.Math`:

```
<xsl:stylesheet version="2.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:fn="http://www.w3.org/2005/xpath-functions">
   <xsl:output method="xml" omit-xml-declaration="yes" />
   <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.

11.1.2.2.2.1 Constructores

Una función de extensión se puede usar para llamar a un constructor .NET. A todos los constructores se les llama con la pseudofunción `new()`. Si hay más de un constructor para una clase, entonces se selecciona el constructor que más se ajusta al número de argumentos suministrados. Si no se encuentra ningún constructor que coincida con los argumentos

suministrados, entonces se genera el error "`No constructor found`".

Constructores que devuelven tipos de datos XPath/XQuery

Si el resultado de una llamada a un constructor .NET se puede convertir de forma implícita en tipos de datos XPath/XQuery, entonces la función de extensión .NET devuelve una secuencia que es un tipo de datos XPath/XQuery.

Constructores que devuelven objetos .NET

Si el resultado de una llamada a un constructor .NET no se puede convertir a un tipo de datos XPath/XQuery adecuado, entonces el constructr crea un objeto .NET contenido con un tipo que es el nombre de la clase que devuelve dicho objeto. Por ejemplo, si se llama al constructor para la clase `System.DateTime` (con `System.DateTime.new()`), entonces se devuelve un objeto que tiene un tipo `System.DateTime`.

Puede que el formato léxico del objeto devuelto no coincida con el formato léxico de un tipo de datos XPath. En estos casos, el valor devuelto (i) debe convertirse al formato léxico del tipo de datos XPath pertinente y (ii) debe convertirse en el tipo de datos XPath necesario.

Se pueden hacer tres cosas con un objeto .NET creado con un constructor:

- Se puede usar dentro de una variable:
```
<xsl:variable name="currentdate" select="date:new(2008, 4, 29)"
xmlns:date="clitype:System.DateTime" />
```
- Se puede pasar a una función de extensión (ver *Métodos de instancia y campos de instancia*):
```
<xsl:value-of select="date:ToString(date:new(2008, 4, 29))"
xmlns:date="clitype:System.DateTime" />
```
- Se puede convertir en un tipo de cadena, numérico o booleano:
```
<xsl:value-of select="xs:integer(data:get_Month(date:new(2008, 4, 29)))"
xmlns:date="clitype:System.DateTime" />
```

11.1.2.2.2.2 Metodos estáticos y campos estáticos

La llamada a un método estático la hace directamente su nombre y se hace presentando los argumentos para el método. El nombre usado en la llamada debe ser el mismo que un método estático público de la clase especificada. Si el nombre del método y el número de argumentos que se dio en la llamada a función coincide con algún método de la clase, entonces los tipos de los argumentos presentados se evalúan para encontrar el resultado ideal. Si no se encuentra ninguna coincidencia, se emite un error.

Nota: un campo de una clase .NET se trata como si fuera un método sin argumentos. Para llamar a una propiedad se usa la sintaxis `get_nombrePropiedad()`.

Ejemplos

Este ejemplo de código XSLT muestra una llamada a un método con un argumento (`System.Math.Sin(arg)`):

```
<xsl:value-of select="math:Sin(30)" xmlns:math="clitype:System.Math"/>
```

Este ejemplo de código XSLT muestra una llamada a un campo (que se trata como si fuera un método sin argumentos) (`System.Double.MaxValue()`):

```
<xsl:value-of select="double:MaxValue()"
xmlns:double="clitype:System.Double"/>
```

Este ejemplo de código XSLT muestra una llamada a una propiedad (la sintaxis es `get_nombrePropiedad()`) (`System.String()`):

```
<xsl:value-of select="string:get_Length('my string')"
xmlns:string="clitype:System.String"/>
```

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>
```

11.1.2.2.2.3 Métodos de instancia y campos de instancia

Un método de instancia es un método al que se le pasa un objeto .NET como primer argumento de la llamada al método. Este objeto .NET se suele crear usando una función de extensión (por ejemplo, una llamada a un constructor) o un parámetro o una variable de una hoja de estilos. Un ejemplo de código XSLT para este tipo de método sería:

```
<xsl:stylesheet version="2.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:fn="http://www.w3.org/2005/xpath-functions">
   <xsl:output method="xml" omit-xml-declaration="yes"/>
   <xsl:template match="/">
      <xsl:variable name="releasedate"
         select="date:new(2008, 4, 29)"
         xmlns:date="clitype:System.DateTime"/>
      <doc>
         <date>
            <xsl:value-of select="date:ToString(date:new(2008, 4, 29))"
               xmlns:date="clitype:System.DateTime"/>
```

```
            </date>
            <date>
               <xsl:value-of select="date:ToString($releasedate)"
                  xmlns:date="clitype:System.DateTime"/>
            </date>
         </doc>
      </xsl:template>
   </xsl:stylesheet>
```

En el ejemplo anterior, se usó un constructor `System.DateTime` (`new(2008, 4, 29)`) para crear un objeto .NET de tipo `System.DateTime`. Este objeto se creó dos veces, una vez como valor de la variable `releasedate`, y otra vez como primer y único argumento del método `System.DateTime.ToString()`. Al método de instancia `System.DateTime.ToString()` se le llama dos veces, ambas con el constructor `System.DateTime` (`new(2008, 4, 29)`) como primer y único argumento. En una de estas instancias, se usó la variable `releasedate` para obtener el objeto .NET.

Métodos de instancia y campos de instancia

La diferencia entre un método de instancia y un campo de instancia es solo teórica. En un método de instancia, se pasa directamente un objeto .NET como argumento. En un campo de instancia, se pasa un parámetro o una variable (aunque el parámetro o la variable puede contener un objeto .NET). Por ejemplo, en el código del ejemplo anterior, la variable `releasedate` contiene un objeto .NET y esta es la variable que se pasa como argumento de `ToString()` en el segundo constructor de elemento `date`. Por tanto, la instancia `ToString()` del primer elemento `date` es un método de instancia, mientras que la segunda se considera un campo de instancia. El resultado es el mismo en ambos casos.

11.1.2.2.2.4 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.

11.1.2.2.2.5 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).

11.1.2.2.3 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.

11.2 Datos técnicos

Esta sección incluye información general práctica sobre aspectos técnicos de su software. Consta de varios apartados:

- Requisitos de OS y memoria
- Analizador XML de Altova
- Motores XSLT y XQuery de Altova
- Compatibilidad con Unicode
- Uso de Internet

11.2.1 Requisitos de SO y memoria

Sistema operativo

Las aplicaciones de software de Altova están disponibles en estas plataformas:

- Windows 7 SP1 con actualización de la plataforma, Windows 8, Windows 10
- Windows Server 2008 R2 SP1 con actualización de la plataforma 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 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.

11.2.2 Validador XML de Altova

Al abrir un documento XML, la aplicación usa su validador integrado para comprobar si el formato XML es correcto, validar el documento con un esquema (si viene dado) y generar árboles e infosets. El validador XML también se usa para ofrecerle ayuda de edición inteligente mientras edita documentos y para mostrar de forma dinámica los errores de validación (si los hay).

El validador XML implementa la recomendación final de la especificación XML Schema 1.0 y 1.1 del W3C. Además Altova actualiza continuamente el validador los avances del grupo de trabajo XML Schema del W3C.

11.2.3 Motores XSLT y XQuery de Altova

Los productos de Altova usan los motores XSLT 1.0, XSLT 2.0, XSLT 3.0, XQuery 1.0 y XQuery 3.0 de Altova. Para más información sobre el comportamiento de cada motor, consulte el anexo *Información sobre motores de Altova.*

Nota: Altova MapForce genera código con los motores XSLT 1.0, XSLT 2.0 y XQuery 1.0.

11.2.4 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.

11.2.5 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.

11.3 Información sobre licencias

En esta sección encontrará:

- información sobre la distribución de este producto de software
- información sobre activación del software y medición de licencias
- información sobre derechos de propiedad intelectual de este producto de software
- 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.

11.3.1 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 antes de decidir si compra el producto.
- 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 totalmente integrado. 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 y aceptar el contrato de licencia de software de Altova, que se entrega en forma de código clave. La licencia puede comprarse directamente en la tienda en línea del sitio web de Altova. Después de comprar la licencia recibirá el código clave, que debe introducir en el cuadro de diálogo "Activación del software" para desbloquear el producto de forma permanente.

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.

Para más información consulte el contrato de licencia de software de Altova que aparece al final de esta sección.

11.3.2 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.

También notará que su producto de Altova ofrece varias funciones prácticas si está conectado a Internet. Estas funciones no tienen nada que ver con la tecnología de medición de licencias.

11.3.3 Derechos de propiedad intelectual

El software de Altova y sus copias (si tiene permiso de Altova para realizar copias) es propiedad intelectual de Altova y de sus proveedores. La estructura, la organización y el código del software se considera secreto comercial e información confidencial de Altova y de sus proveedores. El software está protegido por las leyes de derechos de autor, como la ley de derechos de autor de EE UU, tratados internacionales y la legislación vigente del país donde se utiliza, entre otras. Altova conserva los derechos de propiedad de todas las patentes, derechos de autor, secretos comerciales, marcas registradas y otros derechos de propiedad intelectual pertenecientes al software y los derechos de propiedad de Altova abarcan también imágenes, fotografías, animaciones, vídeos, audio, música, texto y otros applets incorporados al software y al material impreso que viene con el software. Las notificaciones de infracción de dichos derechos de autor debe enviarse al agente de derechos de autor de Altova, cuyos datos de contacto aparecen en el sitio web de Altova.

El software de Altova contiene software de terceros que también está protegido por las leyes de propiedad intelectual, incluida, entre otras, la legislación de derechos de autor mencionada en http://www.altova.com/es/legal_3rdparty.html.

Los demás nombres y marcas registradas son propiedad de sus respectivos propietarios.

11.3.4 Contrato de licencia para el usuario final

- Altova End User License Agreement: http://www.altova.com/eula
- Altova Privacy Policy: http://www.altova.com/privacy

Altova MapForce 2018 Basic Edition

Glosario

12 Glosario

Este glosario incluye una lista de términos utilizados en MapForce.

12.1 A

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 a otro. Una asignación está formada por componentes, que se van añadiendo al área de asignación de MapForce para crear las transformaciones de datos (p. ej. para convertir documentos XML que siguen un esquema en documentos que siguen otro esquema). 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. Los diseños de asignación de MapForce tienen la extensión de archivo **.mfd**.

12.2 C

Componente

En MapForce un componente es el elemento gráfico que representa visualmente la estructura (esquema) de los datos o el elemento que determina cómo se deben transformar los datos (funciones). Los componentes son piezas fundamentales necesarias para construir una asignación. En el área de asignación de MapForce los componentes aparecen en forma de rectángulo. Estos son algunos ejemplos de componente:

- Constantes
- Filtros
- Condiciones
- Funciones
- Documentos EDI (UN/EDIFACT, ANSI X12, HL7)
- Archivos Excel 2007+
- Componentes de entrada simples
- Componentes de salida simples
- Esquemas XML y documentos DTD

Componente de entrada

Un componente de entrada es un componente de MapForce que permite pasar valore simples a una asignación de datos. Los componentes de entrada se suelen utilizar para pasar nombres de archivo y otros valores de cadena a una asignación de datos en tiempo de ejecución. Los componentes de entrada no se deben confundir con los componentes de origen.

Componente de salida

Un componente de salida (o de salida simple) es un componente de MapForce que permite recuperar un valor de cadena de la asignación de datos. Los componentes de salida son uno de los muchos tipos de componente de destino, pero no son sinónimos.

Componente de origen

Un componente de salida es un componente donde MapForce lee datos. Al ejecutar la asignación, MapForce lee lo datos que aporta el conector del componente de origen, los convierte al tipo elegido y los envía al conector del componente de destino.

Componente de destino

Un componente de destino es un componente donde MapForce escribe datos. 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. El componente de destino es lo opuesto de un componente de origen.

Componente de combinación

Un componente de combinación es un componente de MapForce que permite unir varias estructuras en la asignación de acuerdo con las condiciones definidas por el usuario. Devuelve la asociación (conjunto combinado) de elementos que cumplen con la condición. Las combinaciones son de gran ayuda a la hora de combinar datos de dos estructuras que tienen un campo en común (p. ej. una identidad).

Conexión

Una conexión es la línea que se puede dibujar para unir dos conectores. Al dibujar la línea estamos dando a MapForce instrucciones de transformar lo datos de una forma determinada (p. ej. leer datos de un documento XML y escribirlos en otro documento XML).

Conector

Un conector es un pequeño triángulo situado a la izquierda o derecha de un componente. Los conectores situados a la izquierda del componente ofrecen puntos de entrada para el componente. Los situados a la derecha ofrecen puntos de salida desde el componente.

Contexto primario (parent-context)

parent-context es un argumento opcional de algunas funciones de agregado de la biblioteca de funciones **core** de MapForce (como las funciones `min`, `max`, `avg` y `count`). En un componente de origen que tiene varias secuencias jerárquicas, el contexto primario determina en qué conjunto de nodos debe operar la función.

12.3 F

FLF (*Fixed Length Field* o campo de longitud fija)

Frecuente formato de texto donde los datos se separan por campos de longitud fija (p. ej. los primeros cinco caracteres de cada fila representan un ID de transacción y los 20 caracteres siguientes representan una descripción de la transacción).

FlexText

FlexText es un módulo de MapForce Enterprise Edition que sirve para convertir datos de archivos de texto heredados y no estándar de gran complejidad en otros formatos compatibles con MapForce y viceversa.

12.4 M

MapForce

MapForce es un entorno IDE multiuso basado en Windows que sirve para transformar datos de un formato en otro o pasar datos de un esquema a otro mediante operaciones visuales de arrastrar y colocar. Para ello ofrece una interfaz gráfica donde el usuario no necesita crear código de programación. De hecho, MapForce genera automáticamente el código de programa necesario para realizar la transformación de datos propiamente dicha (o la asignación de datos). Si prefiere no generar código de programa, puede simplemente ejecutar la transformación con el lenguaje de transformación integrado de MapForce (disponible en las ediciones Professional y Enterprise).

MFF

Extensión de archivo de los archivos de funciones de MapForce.

MFD

Extensión de archivo de los documentos de diseño de MapForce (asignaciones de datos).

12.5 R

Recursos globales

Los recursos globales de Altova es una característica que permite hacer referencia a archivos, carpetas y bases de datos y poder reutilizar este tipo de recursos, configurarlos y ponerlos a disposición de otras aplicaciones de Altova.

Índice

A

B

C

D

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z