Altova MapForce 2024 Basic Edition

Manuel de l'utilisateur et de référence

Altova MapForce 2024 Basic Edition
Manuel de l'utilisateur et de référence

Published: 2024

Table des matières

1 Introduction

Altova MapForce 2024 Basic Edition est un outil de transformation des données des données et ETL puissant pour l'intégration des données. MapForce est une application Windows 32/64-bit qui est exécutée sur Windows 10, Windows 11, et Windows Server 2016 ou plus récent. La prise en charge 64-bit est disponible pour les éditions Enterprise et Professional.

MapForce vous permet de convertir les données depuis et vers presque tout format. MapForce a une interface graphique 20 qui inclut de nombreuses options pour gérer, visualiser, manipuler et exécuter des mappages individuels et des projets de mappage complexes. Pour la transformation des données, MapForce fournit une grande bibliothèque de traitement des données et fonctions de conversion 194 pour filtrer et manipuler les données selon les besoins de votre projet d'intégration des données.

Dès que vous avez terminé de concevoir votre mappage, vous pouvez visualiser la sortie dans un volet séparé et enregistrer la sortie dans l'emplacement désiré. Vous pouvez aussi générer du code 65 pour l'exécution externe.

Vous pouvez élargir la fonction MapForce en intégrant MapForce avec d'autres produits Altova :

- Vous pouvez exécuter vos mappages utilisant le serveur MapForce. Ceci vous aidera à automatiser les opérations professionnelles qui requièrent des transformations de données répétitives. MapForce Server inclura un moteur de transformation des données et peut réaliser la conversion des données any-to-any. De manière plus importante, il s'agit d'un serveur toute plate-forme disponible sur Windows, macOS et Linux.
- RaptorXML Server est un moteur hyper rapide qui valide vos instances.
- FlowForce Server vous aide à automatiser vos tâches et vous permet d'exécuter vos mappages comme tâches prévues.
- StyleVision Server génère l sortie dans HTML, RTF, PDF et Word.
- Vous pouvez utiliser StyleVision pour concevoir des feuilles de style StyleVision Power qui permettent à StyleVision Server de générer la sortie en de multiples formats.
- DatabaseSpy est un outil polyvalent qui vous permet de concevoir, éditer et interroger des bases de données.
- XMLSpy est particulièrement utile si vous voulez éditer les fichiers de mappage. Certains dialogues MapForce vous permettent d'ouvrir les fichiers directement dans XMLSpy.
- Vous pouvez aussi utiliser MapForce comme plug-in de Microsoft Visual Studio et Eclipse. Ceci vous permet d'accéder à la fonction de MapForce sans quitter votre environnement de développement préféré.

Dernière mise à jour : 9 April 2024

1.1 Nouvelles fonctions

Cette section décrit les nouvelles fonctions de chaque release de MapForce. Pour plus de détails, veuillez voir la sous-section respective.

1.1.1 Version 2024

Version 2024 Release 2

- Il est désormais possible d'accéder à divers tutoriels vidéo dans le projet **`MapForceExamples`** (*éditions Professional et Enterprise*). De plus, vous pouvez ajouter vos propres liens aux ressources externes.
- Quand vous déployez votre mappage vers FlowForce Server, vous pouvez choisir de joindre les fichiers de mappage pour une extraction ultérieure. Ceci vous permettra d'éviter la perte de vos fichiers de mappage et vous permettra de les télécharger à tout moment (*Professional and Enterprise editions*).
- Les composants de base de données peuvent désormais partager la même connexion de base de données au moment de l'exécution (*éditions Professional et Enterprise*).
- Il est désormais possible de créer un mappage de valeur des types d'énumération dans XML (*toutes éditions*) et composants XBRL (*Enterprise Edition*). Cette fonction rend le mappage des valeurs d'énumération plus facile et plus rapide : Les deux côtés du Value-Map deviennent « pre-filled » avec toutes les valeurs d'énumération et vous n'aurez qu'à revoir et éditer les valeurs pertinentes de Value-Map. Pour plus d'informations, voir Value-Maps 186.
- Prise en charge de .NET 8.0 pour la génération de code C# (*éditions Professional et Enterprise*). Pour les détails, voir Génération de code 65.
- Prise en charge des message pour FORTRAS EDI (*Enterprise Edition*).
- Prise en charge pour PostgreSQL 16, MySQL 8.2., MySQL 8.3, MariaDB 11.2, SQLite 3.45 (*éditions Professional et Enterprise*).
- Mises à jour internes et optimisations.

Version 2024

- Une nouvelle fonctionnalité de MapForce appelée Extracteur PDF est désormais disponible (*Enterprise Edition*). L'Extracteur PDF vous permet de créer des modèles d'extraction PDF que vous pouvez importer dans MapForce et utiliser comme composants source dans vos mappages.
- Il est désormais possible de créer des mappages alimentés par l'IA dans MapForce (*Enterprise Edition*). MapForce vous permet de créer des appels de service REST dans une API, tels que OpenAI API, Azure OpenAI API, AWS AI Services, etc.
- Prise en charge pour SWIFT 2023 (*Enterprise Edition*).
- Une nouvelle fonction **`sleep`** est désormais disponible, qui permet de passer des données après un délai spécifié (*Professional et Enterprise Edition*).
- Une prise en charge native a été ajoutée pour MySQL et MariaDB (*éditions Professional et Enterprise*).
- La fonction des connexions d'enfants correspondants a été améliorée et élargie pour inclure de nouvelles options correspondantes. Pour les détails, voir Connexions d'enfants correspondants 52.
- Les types Grant *Identifiants Client* et *Identifiants du Mot de passe du propriétaire de la Ressource* sont désormais pris en charge dans les identifiants OAuth, en plus du type Grant *Code d'autorisation* (*Enterprise Edition*).
- Mises à jour internes et optimisations.

1.1.2 Version 2023

Version 2023 Release 2

- Il est désormais possible de générer la déclaration `standalone="yes"` dans la déclaration XML des fichiers cible XML. Pour les détails, voir Paramètres de composant XML[116].
- Le système Aide[478] a été réorganisé pour fournir une Aide en ligne par défaut, avec une option pour utiliser le manuel utilisateur PDF installé localement[469] comme défaut alternatif.
- Il est désormais possible d'ajouter des commentaires « sticky-note-style » à un mappage. Pour plus d'information, voir Commentaires[34].
- Les paramètres ont été ajoutés pour définir les paramètres de réseau[473].
- Une prise en charge des messages VDA EDI a été ajoutée (*Enterprise Edition*).
- Mises à jour internes et optimisations.

Version 2023

- Une prise en charge pour les thèmes suivants a été ajoutée : *Classique*, *Clair* et *Sombre*. Pour plus d'information, voir Fenêtre[477].
- Mises à jour internes et optimisations.
- La prise en charge d'Eclipse a été mise à jour et couvre maintenant les versions suivantes : 2022-09, 2022-06, 2021-03, 2020-12 (*éditions Professional et Enterprise*).
- Prise en charge des message pour ODETTE EDI (*Enterprise Edition*).
- Prise en charge de XII Transformation Registry 5 Specification (*Enterprise Edition*).
- Il est désormais possible de créer des paramètres UDF[210] basés sur base de données et de variables[159] avec une arborescence de tables associées (*éditions Professional et Enterprise*).
- Il est désormais possible d'envoyer une structure de requête `application/x-www-form-urlencoded` au service REST (*Enterprise Edition*).
- Prise en charge des Répertoires UN/EDIFACT D.21B et D.22A Directories (*Enterprise Edition*).
- Prise en charge pour SQLite 3.39.2, MariaDB 10.9.2 et PostgreSQL 14.5 (*éditions Professional et Enterprise*).
- Prise en charge du Gestionnaire de schéma XML[129] est un outil qui propose un moyen centralisé d'installer et de gérer des schémas XML pour une utilisation sur toutes les applications activées par XBRL d'Altova.
- Prise en charge des délimiteurs EDI mappables (*Enterprise Edition*). La fonction est actuellement prise en charge pour les normes EDI suivantes : EDIFACT, X12 et NCPDP SCRIPT.

1.1.3 Version 2022

Version 2022 Release 2

- Mises à jour et optimisations internes
- La prise en charge d'Eclipse a été mise à jour et couvre maintenant les versions suivantes : 2021-12, 2021-09; 2021-06; 2021-03 (*éditions Professional et Enterprise*).
- Prise en charge de Visual Studio 2022 dans MapForce Plug-in pour Visual Studio et génération de code (*éditions Professional et Enterprise*).
- Prise en charge de .NET 6.0 dans génération de code (*éditions Professional et Enterprise*).

- De nouvelles versions de base de données sont prises en charge : PostgreSQL 14, SQLite 3.37.2, MariaDB 10.6.5, MySQL 8.0.28, IBM DB2 11.5.7 (*éditions Professional et Enterprise*).
- Il est désormais possible de prévisualiser des images dans la fenêtre **Projet**(*éditions Professional et Enterprise*).
- Il est désormais possible de créer des indicateurs de classement pour des composants XBRL cible (*Enterprise Edition*) conformes à EBA.

Version 2022

- Mises à jour et optimisations internes
- La prise en charge d'Eclipse a été mise à jour et couvre maintenant les versions suivantes : 2021-09; 2021-06; 2020-03; 2020-12 (*éditions Professional et Enterprise*).
- Copy-all connections 55 prend désormais en charge JSON. Cette fonction est uniquement disponible pour les types JSON compatibles (*Enterprise Edition*).
- Un nouveau volet de sortie StyleVision dénommé *Texte* a été introduit. Si un fichier SPS est annexé à un composant, le nouveau format texte brut peut être vu dans MapForce (*éditions Professional et Enterprise*).
- Prise en charge de JSON Schema dans les variables 158 et paramètres UDF 209 (*Enterprise Edition*).
- Prise en charge pour bases de données NoSQL : MongoDB et CouchDB (*Enterprise Edition*).
- Une nouvelle bibliothèque de fonction bson est désormais disponible, qui vous permet de créer et manipuler quelques-uns des types BSON (*Enterprise Edition*).
- Prise en charge des Répertoires UN/EDIFACT D.20B et D.21A.
- Prise en charge de SWIFT 2021.

1.1.4 Version 2021

Version 2021 Release 3

- Prise en charge pour le nouveau JSON Schema Draft 2019-09 et Draft 2020-12 (*uniquement Enterprise Edition*).

Version 2021 Release 2

- XSLT 3.0 est désormais pris en charge en tant que langage de mappage. Voir Générer le code XSLT 65. MapForce inclut désormais également de nouvelles fonctions built-in qui sont prises en charge quand le langage de mappage est XSLT 3.0. Pour plus d'informations, voir Référence des bibliothèques de fonctions 232.
- Mises à jour internes et optimisations.

Version 2021

- Mises à jour internes et optimisations.

1.1.5 Version 2020

Version 2020 Release 2

- Une nouvelle fenêtre Gérer la fenêtre des bibliothèques[23] est disponible qui vous permet de consulter et de gérer toutes les bibliothèques de fonction importées au niveau de document et de programme (cela comprend les fonctions définies par MapForce et d'autres types de bibliothèques). Cela vous permet, par exemple, de copier-coller aisément des fonctions définies par l'utilisateur d'un mappage à un autre, voir Copier-coller des UDF entre les mappages[208].
- Lorsqu'un fichier de mappage importe des bibliothèques, le chemin des fichiers de bibliothèque importée est relatif au fichier de mappage par défaut, voir Chemins de bibliothèque relatifs[201]. Vous pouvez toujours importer des mappages au niveau de l'application, comme dans les releases précédentes, mais dans ce cas, le chemin de bibliothèque est toujours absolu.
- Si un fichier de mappage importe des bibliothèques XSLT, vous pouvez générer un code XSLT qui référence les fichiers de bibliothèque importés en utilisant un chemin relatif. La nouvelle option est disponible dans le dialogue Paramètres de mappage[74].
- Mises à jour et optimisations internes

Version 2020

- Lorsque vous souhaitez remplacer des valeurs avec une table de consultation, vous pouvez coller des données tabulaires (paires key-value) depuis des sources externes comme CSV ou Excel dans le mappage. De même, il est plus facile de gérer des cas lorsqu'une valeur n'est pas trouvée dans la table de consultation prédéfinie, le traitement de ce type de valeurs ne nécessite plus l'utilisation de la fonction **`substitute-missing`**. Voir Utiliser Value-Maps[182].
- Mises à jour et optimisations internes

1.2 MapForce, c'est quoi?

Site web d'Altova : Outil de mappage de données

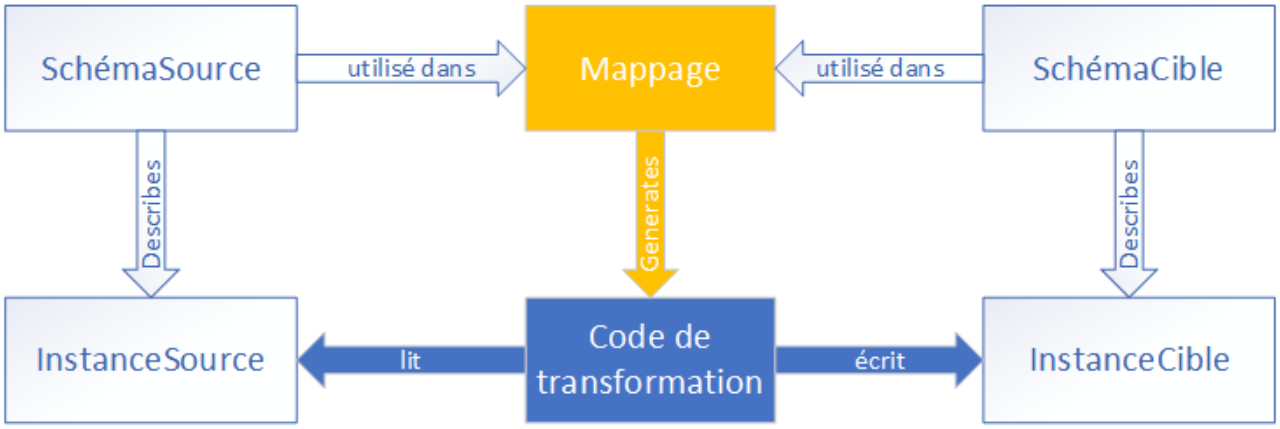
MapForce est un outil graphique puissant pour la conversion et l'intégration any-to-any. Voir Mappage : Sources et Cibles 16 pour une liste complète de formats de données disponibles. Un mappage typique consiste en une ou plusieurs sources de données et une ou plusieurs cibles de données 32. Le mappage peut aussi inclure un ou plusieurs composants de transformation 33 qui fournissent une large gamme de traitement des données et d'options de filtrage. Pour en savoir plus sur divers scénarios de mappage, voir Scénarios de mappage 18 et Tutoriels 77.

Afin réaliser un mappage, vous pouvez fournir une structure de données qui décrit la structure de chacun de vos fichiers source et cible. Par exemple, un schéma XML définit la structure d'un document XML. Le mappage (de la source à la cible) est réalisé par le biais d'une interface utilisateur graphique glisser-déplacer. Vous n'êtes pas obligé d'écrire un code de programme pour le mappage. MapForce vous génère le code. Vous pouvez ensuite utiliser ce code pour transformer les documents avec la structure de données source aux documents avec une structure de données cible.

Toutes les éditions de MapForce sont disponibles en tant qu'applications 32-bit. Les éditions MapForce Professional et Enterprise sont également disponibles en tant qu'applications 64-bit.

Modèle abstrait

Le modèle abstrait ci-dessous illustre un des scénarios de base d'une transformation de données dans MapForce. Le schéma source décrit la structure de l'instance source. Le schéma cible décrit la structure de l'instance cible. Dépendant de vos besoins, les schémas source et cible peuvent être la même ou une autre structure. Quand vous connectez la source et la cible, le mappage génère le code de transformation (dans le langage sélectionné 17 sélectionné) qui lit les données de l'instance source et écrit ces données dans l'instance cible. Pour voir comment ce modèle de transformation des données est mis en œuvre dans un exemple concret, voir le Tutoriel 1 78.



Dans des situations « real-life », vous pouvez mélanger apparier toute combinaison de sources de données (par ex., XML, EDI et fchiers texte) et les mapper avec toute combinaison de cibles de données (par ex., la base de données et le fichier Excel).

Conventions

Les fichiers de mappage illustrés et référencés dans le manuel peuvent être trouvés sous les emplacements suivants :

- `C:\Users\<username>\Documents\Altova\MapForce2024\MapForceExamples`
- `C:\Users\<username>\Documents\Altova\MapForce2024\MapForceExamples\Tutorial`
- `C:\Users\<username>\Documents\Altova\MapForce2024\MapForceExamples\Tutorial\BasicTutorials`

Dans cette section

Cette section est organisée dans les rubriques suivantes :

1.2.1 Mappage : Sources et cibles

Dans MapForce, les termes *source* et *cible* sont des termes essentiels qui se réfèrent à des structures de données desquels ou vers lesquels les données sont mappées, respectivement. Les technologies qui peuvent être utilisées en tant que sources et cibles de mappage sont recensées ci-dessous.

MapForce Basic Edition

- XML et Schéma XML

MapForce Professional Edition

- XML et Schéma XML
- Fichiers plats, y compris des valeurs séparées par virgule (CSV) et un format de champ à longueur fixe (FLF) ;
- Bases de données : toutes les bases de données relationnelles majeures
- Fichiers binaires (contenu BLOB brut)

MapForce Enterprise Edition

- XML et Schéma XML
- Fichiers plats, y compris des valeurs séparées par virgule (CSV) et un format de champ à longueur fixe (FLF) ;
- Les données provenant des fichiers de texte de legacy peuvent être mappées et converties sur d'autres formats avec MapForce FlexText
- Bases de données SQL : toutes les bases de données relationnelles majeures
- Bases de données NoSQL
- Fichiers binaires (contenu BLOB brut)
- Standards EDI
- Fichiers JSON

- Microsoft Excel 2007 et les fichiers ultérieurs
- Fichiers d'instance XBRL et taxonomies
- Protocol Buffers
- Les fichiers PDF basés sur des modèles PDF créés dans l'Extracteur PDF (peut uniquement être utilisée comme sources de données)

1.2.2 Langages de transformation

Dans MapForce, un langage de transformation est utilisé pour générer le code de transformation qui exécute les mappages. Vous pouvez sélectionner/modifier un langage de transformation à tout moment. Vous pouvez générer le code de programme via la commande de menu **Fichier | Générer Code dans** ou **Fichier | Générer Code dans langage sélectionné** et utilisez ce code pour exécuter des transformations de données à l'extérieur de MapForce. Pour plus d'information, voir Génération de code 65.

Dépendant de l'édition MapForce, vous pouvez choisir la langue préférée pour vos transformations de données comme suit :

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

Si vous sélectionnez XSLT 1-3 ou XQuery comme langage de transformation, vous serez à même de consulter le code de transformation dans un volet séparé de MapForce.

Pour sélectionner un langage de transformation, suivez les étapes suivantes :

- Dans le menu **Sortie**, cliquez sur le nom que vous souhaitez utiliser pour la transformation.
- Cliquez sur le nom du langage dans la barre d'outils **Sélection du langage** (*affichée ci-dessous*).

Lorsque vous changez de langage de transformation du mappage, certaines fonctions de MapForce ne sont pas prises en charge pour ce langage. Pour plus d'information, voir Notes de prise en charge 484.

Pendant que vous concevez ou visualisez des mappages, MapForce valide l'intégrité de vos schémas et transformations. Si une erreur de validation devait apparaître, MapForce l'affiche dans la fenêtre des Messages 25. Ceci est très utile , car vous pouvez immédiatement la réviser et corriger ces erreurs.

BUILT-IN
Quand vous sélectionnez Built-In comme langage de transformation, MapForce utilise son moteur de transformation natif pour exécuter les mappages. MapForce utilise également cette option de manière implicite

à chaque fois que vous visualisez la sortie d'un mappage dont le langage de transformation est Java, C#, ou C++.

Le moteur Built-In exécute les mappages sans avoir besoin de processeurs externes, ce qui pourrait être un bon choix si l'utilisation de la mémoire vous cause des soucis. Si vous n'avez pas besoin de générer le code de programme dans un langage spécifique, utilisez Built-In comme option par défaut car il prend en charge la plupart des fonctions de MapForce comparé à d'autres langages (voir Notes de prise en charge 484). De plus, si vous sélectionnez Built- avant comme langage de transformation, vous serez en mesure d'automatiser le mappage avec MapForce Server. Pour plus d'informations, voir Automatisation avec les produits Altova 425.

1.2.3 Scénarios de mappage

Site web d'Altova : MapForce, les vidéos démo

Dépendant de vos besoins et exigences professionnels, la complexité de vos mappages peut varier : Par exemple, vous allez éventuellement devoir configurer votre mappage pour lire les données d'une source et écrire ces données dans de multiples cibles ou fusionner des données de multiples sources dans une cible. Différentes structures de données peuvent être utilisées en tant que sources et cibles : par exemple, fichiers XML, un fichier XML, des bases de données, des fichiers EDI, etc. Pour en savoir plus sur les formats acceptables des sources et cibles, voir Mappage : Sources et cibles 16.

La complexité des designs de mappage est illustrée dans les scénarios suivants mais pas limitée à ceux-ci :

- Mapper une source vers une cible Pour plus d'informations, voir Tutoriel 1 78.
- Fusionner des sources de données multiples dans une cible. Pour plus d'informations, voir Tutoriel 2 89.
- Mapper les données d'une source à la première cible, puis filtrer les données de telle manière que seul un sous-ensemble de ces données est mappé à la deuxième cible. Voir le Tutoriel 3 94.
- Mapper les sources multiples vers les cibles multiples. Voir le Tutoriel 4 102.

Peu importe la technologie avec laquelle vous travaillez, MapForce détermine généralement automatiquement la structure de vos données ou suggère la fourniture d'un schéma pour vos données. MapForce peut également générer des schémas depuis un exemple de fichier d'instance. Par exemple, si avez un fichier d'instance XML mais pas de définition de schéma, MapForce peut la générer pour vous. MapForce rend donc les données à l'intérieur des fichiers XML disponibles pour le mappage vers d'autres fichiers ou formats. Pour en savoir plus sur les termes et fonctions basiques de MapForce, voir Notions fondamentales de mappage 30 et Aperçu de l'interface d'utilisateur 20.

Projets (éditions Professional et Enterprise)
Pour un accès et une gestion plus faciles, vous pouvez organiser les designs de mappage dans les projets de mappage. En plus de la génération de code pour les mappages individuels au sein du projet, vous pouvez générer le code de programme depuis des projets entiers.

1.2.4 Intégration avec les produits d'Altova

Les transformations peuvent être exécutées au sein de MapForce en utilisant les moteurs built-in XSLT/XQuery. MapForce peut également être utilisé en tandem avec d'autres produits d'Altova (*voir ci-dessous*).

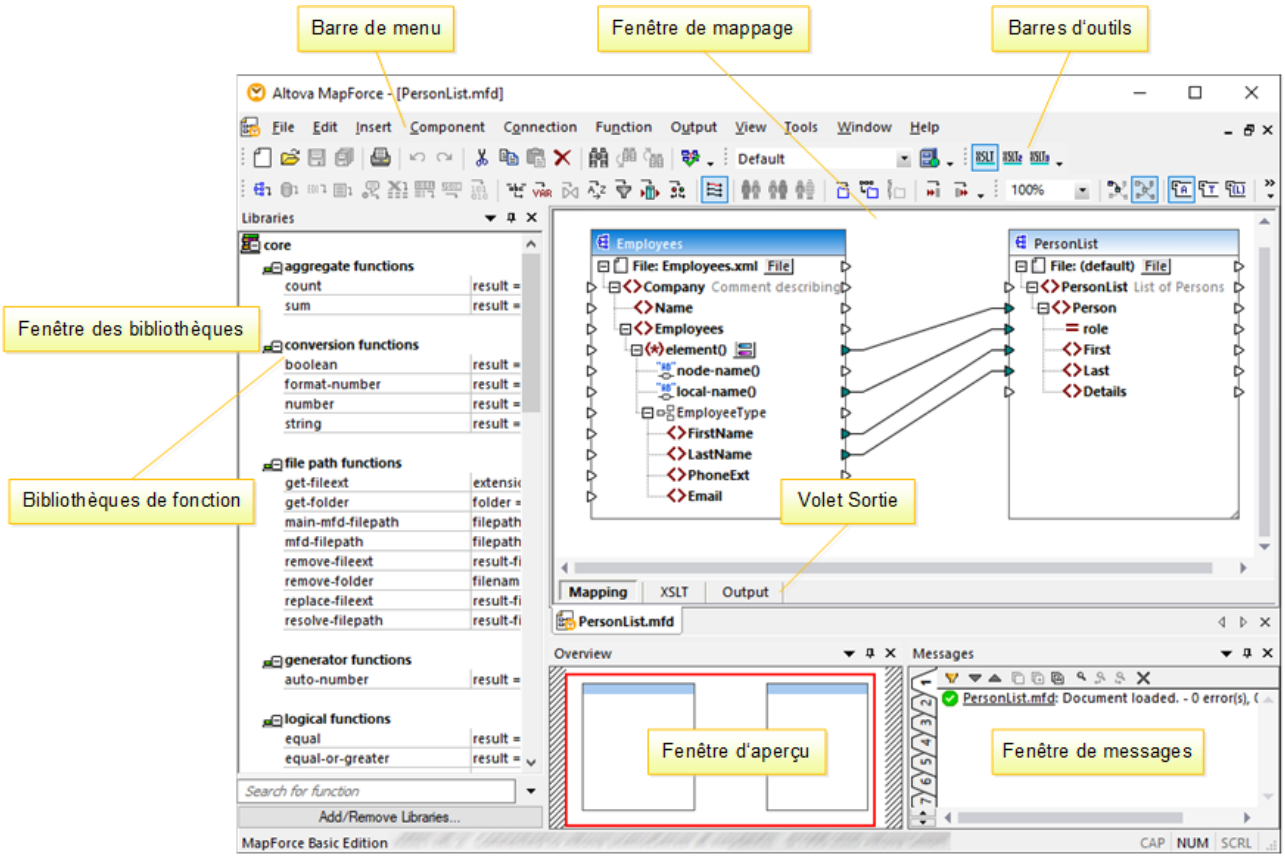
XMLSpy
Si XMLSpy est installé sur le même appareil, vous pouvez ouvrir et éditer tout type de fichier pris en charge en ouvrant XMLSpy directement depuis les contextes spécifiques de MapForce. Par exemple, la commande de menu **Composant | Éditer une définition de schéma dans XMLSpy** est disponible quand vous cliquez sur une composante XML.
RaptorXML Server
Vous pouvez choisir d'exécuter la XSLT générée directement dans MapForce et de consulter le résultat de transformation des données immédiatement. Si vous nécessitez une performance accrue, vous pouvez traiter le mappage en utilisant RaptorXML Server, un moteur de transformation XML extrêmement rapide.
MapForce Serveur (Éditions Enterprise et Professional)
Vous pouvez automatiser les tâches de MapForce avec l'aide de Altova MapForce Server, qui peut être installé sur Windows, Linux et les systèmes macOS. MapForce Server vous permet d'exécuter les transformations spécifiées dans un mappage, non seulement depuis la ligne de commande du système d'exploitation respectif mais aussi à travers les appels d'API (.NET, COM, Java).
FlowForce Serveur (Éditions Enterprise et Professional)
Vous pouvez également automatiser les tâches de MapForce avec l'aide de Altova FlowForce Server, qui peut être installé sur Windows, Linux et les systèmes macOS. FlowForce Server vous permet d'exécuter des tâches de MapForce Server conformément à un calendrier.
StyleVision (Éditions Enterprise and Professional)
Avec l'aide de StyleVision, vous pouvez concevoir ou réutiliser des feuilles de style StyleVision Power existantes et consulter le résultat des transformations de mappage en tant que documents HTML, RTF, PDF ou Word 2007+.
MapForce en tant que plug-in
Les éditions MapForce Professional et Enterprise peuvent être installées en tant que plug-in des environnements de développement intégrés de Visual Studio et Eclipse. Ainsi, vous pouvez concevoir des mappages et obtenir l'accès aux fonctions MapForce sans quitter votre environnement de développement préféré.

Pour plus d'information sur l'automatisation des tâches, voir Automatiser les tâches de MapForce avec les produits d'Altova 425.

1.3 Aperçu de l'interface d'utilisateur

L'interface graphique d'utilisateur de MapForce est organisée comme un environnement de développement intégré. Les principaux composants d'interface sont illustrés ci-dessous. Vous pouvez changer les paramètres d'interface en utilisant la commande de menu **Outils | Personnaliser**. Utilisez les boutons affichés dans le coin du haut à droite de chaque fenêtre pour les afficher, masquer, épingler ou ancrer. Si vous devez rétablir les barres d'outils et les fenêtres à leur état par défaut, utilisez la commande de menu **Outils | Restaurer les barres d'outils et les fenêtres**.

L'image ci-dessous illustre les parties principales de l'interface utilisateur graphique de MapForce.



Pour plus d'information sur les fonctionnalités et fonctions de chaque partie de l'interface, voir la page ci-dessous :

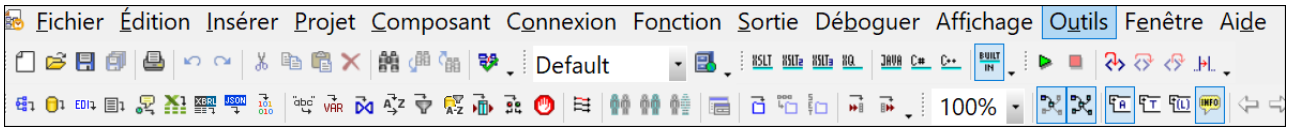
- Barres 21
- Fenêtres 21
- Fenêtre de messages 25
- Volets 26

1.3.1 Barres

Cette rubrique donne un aperçu des barres disponibles.

Barre de menu et barre d'outils

La barre de **menu** affiche les items de menu. Chaque barre d'outils affiche un groupe de touches représentant les commandes de MapForce. Vous pouvez repositionner les barres d'outils en glissant leur poignée à l'endroit désiré. La capture d'écran ci-dessous illustre la barre de **menu** et les barres d'outils. L'interface actuelle dépend de votre édition de MapForce et des paramètres que vous choisissez.



Barre de statut d'application

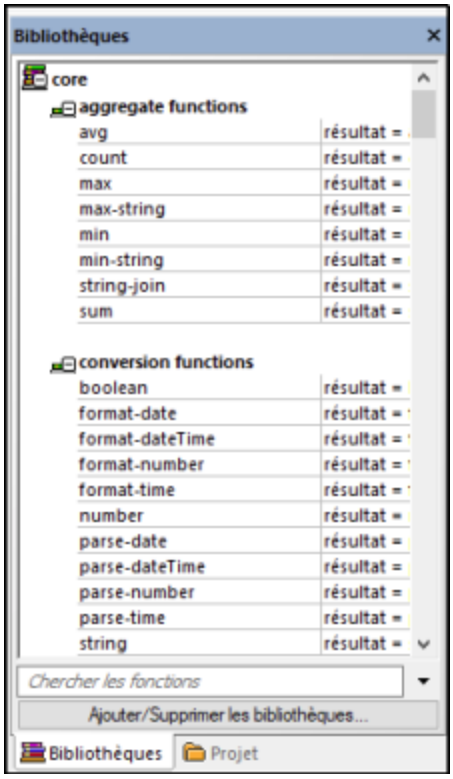
La barre de statut d'application apparaît en bas de la fenêtre d'application et montre l'information au niveau de l'application. Les info-bulles sont affichées quand vous déplacez la souris au-dessus d'un bouton de barre d'outils. Si vous utilisez la version 64-bit de MapForce, le nom d'application apparaît dans la barre de statut avec le suffixe (x64). Il n'y a pas de suffixe pour la version 32-bit.

1.3.2 Fenêtres

Cette rubrique donne une vue d'ensemble des fenêtres disponibles.

Fenêtre de bibliothèques

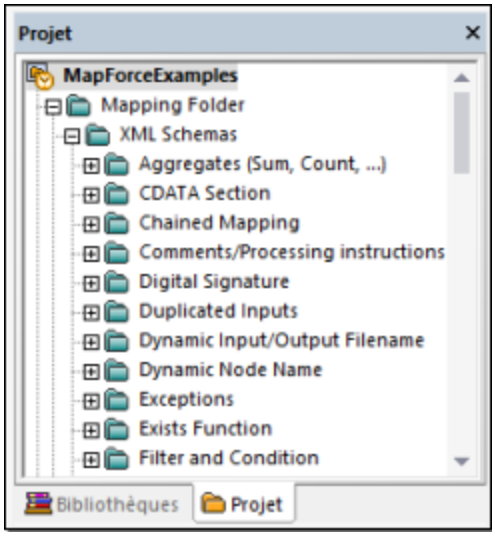
La fenêtre **Bibliothèques** recense les fonctions intégrées MapForce, qui sont organisées en bibliothèques. La liste des fonctions disponibles change sur la base du langage de transformation que vous avez choisi soit depuis le menu **Sortie**, soit depuis la barre d'outils **Sélection de langage**. Pour plus d'information, voir Langages de transformation 17. Si vous avez créé des fonctions définies par l'utilisateur, ou si vous avez importé des bibliothèques externes, elles apparaîtront aussi dans la fenêtre des **Bibliothèques**.



Pour chercher des fonctions par leur nom ou leur description, saisissez la valeur de recherche dans le champ de saisie situé en bas de la fenêtre **Bibliothèques**. Pour trouver toutes les occurrences d'une fonction (dans le mappage actif actuellement), cliquez avec la touche de droite sur la fonction et choisissez **Trouver tous les appels** depuis le menu contextuel. Vous pouvez aussi consulter le type de données de fonction et la description directement depuis la fenêtre **Bibliothèques**. Pour plus d'informations, voir Fonctions 194.

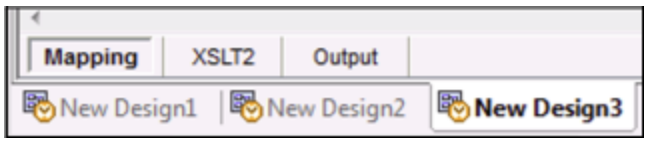
Fenêtre Projet (Éditions Enterprise et Professional)

MapForce prend en charge l'Interface de document multiple et permet de regrouper vos mappages dans des projets de mappage. La fenêtre **Projet** montre tous les fichiers et les dossiers qui ont été ajoutés au projet. Les fichiers de projet ont l'extension **`*.mfp`** (MapForce Project). Pour rechercher des mappages au sein des projets, cliquez n'importe où au sein de la fenêtre **Projet** et appuyez sur **CTRL + F**.



Fenêtre(s) de mappage

MapForce utilise une Multiple Document Interface (MDI). Chaque fichier de mappage que vous ouvrez dans MapForce a une fenêtre séparée. Cela vous permet de travailler avec plusieurs fenêtres de mappage et d'arranger ou de redimensionner les fenêtres de plusieurs manières dans la fenêtre principale (parent) de MapForce. Vous pouvez aussi arranger toutes les fenêtres ouvertes en utilisant les mises en page Windows standard : Disposer horizontalement, Disposer verticalement, Cascade. Lorsque de multiples mappages sont ouverts dans MapForce, vous pouvez passer rapidement entre les onglets disposés dans la partie inférieure du volet **Mappage** (*voir la capture d'écran ci-dessous*).



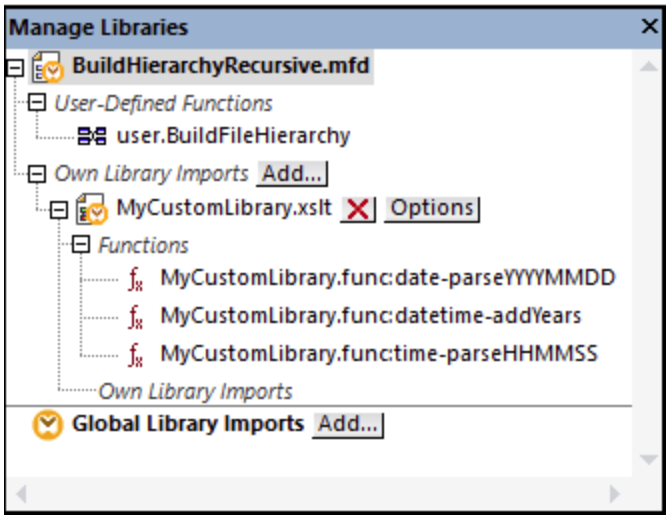
Vous pouvez ouvrir les options de gestion de la Fenêtre en utilisant la commande de menu **Fenêtre | Fenêtres**. Le dialogue **Fenêtres** vous permet de réaliser différentes actions y compris activer, enregistrer, fermer ou minimiser les fenêtres de mappage ouvertes. Pour choisir plusieurs fenêtres dans le dialogue **Fenêtres**, cliquez sur les entrées nécessaires tout en maintenant la touche **Ctrl** appuyée.

Gérer la fenêtre Bibliothèques

Cette fenêtre vous permet de consulter et de gérer toutes les fonctions définies par l'utilisateur (FDU) et les bibliothèques personnalisées importées qui sont utilisés par les mappages actuellement ouverts.

Par défaut, la fenêtre Gérer des bibliothèques n'est pas visible. Pour l'afficher, choisir une des deux options suivantes :

- Dans le menu **View**, cliquer sur **Gérer Bibliothèques**.
- Cliquer **Ajouter/Supprimer Bibliothèques** en bas de la fenêtre Bibliothèques.



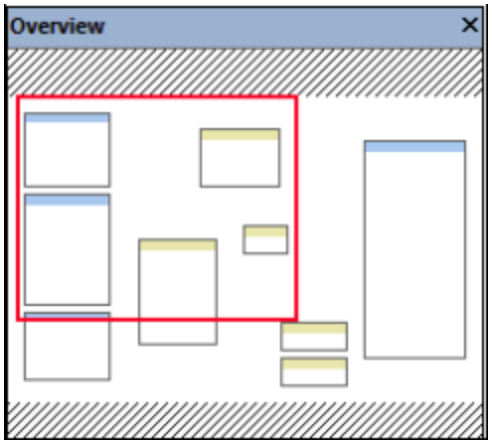
Vous pouvez choisir de consulter des UDF et des bibliothèques uniquement pour le document de mappage (fichier .mfd) qui est activé actuellement, ou pour tous les mappages ouverts. Pour consulter des fonctions et des bibliothèques importées pour tous les documents de mappages ouverts actuellement, cliquer avec la touche de droite dans la fenêtre et sélectionner **Afficher les documents ouverts** depuis le menu contextuel.

Pour afficher le chemin du document de mappage ouvert au lieu du nom, cliquer avec la touche de droite dans la fenêtre et sélectionner **Afficher les chemins de fichier** depuis le menu contextuel.

Pour plus d'informations, voir Gérer les Bibliothèques de fonction 198.

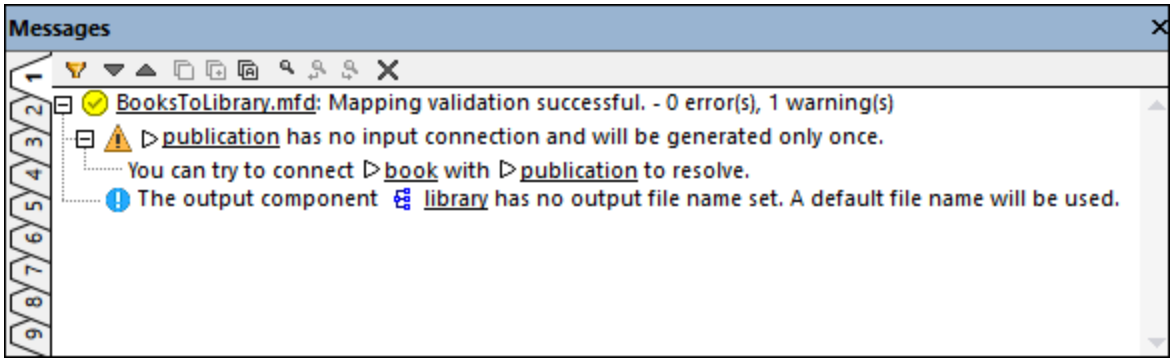
Fenêtre de vue d'ensemble

La fenêtre **Aperçu** vous donne une vue d'ensemble du volet de mappage 26. Utilisez-la pour naviguer rapidement vers un emplacement particulier dans la zone de mappage lorsque la taille de mappage est très importante. Pour vous rendre à un emplacement particulier sur le mappage, cliquez et glissez le rectangle rouge.



1.3.3 Fenêtre de messages

La fenêtre de **Messages** (*voir la capture d'écran ci-dessous*) affiche des statuts de de validation, des messages, des erreurs et/ou des avertissements lorsque vous consultez ou validez 63 un mappage. Cliquez sur le texte souligné dans la fenêtre de **Messages** pour voir un composant ou une structure qui a causé le message d'information, d'avertissement ou d'erreur.



Valider l'icône du statut de validation

Lorsque vous validez un mappage, MapForce vérifie, par exemple, des sortes de composants non pris en charge et des connexions incorrectes ou manquantes. Le résultat de la validation est affiché dans la fenêtre de **Messages** avec une des icônes de statut suivantes :

Icône	Signification
	Validation a été achevée avec succès.
	Validation a été achevée avec des avertissements.
	Validation a échoué.

La fenêtre **Messages** peut afficher tout type de message supplémentaire suivant : messages d'information, avertissements et erreurs.

Icône	Signification
	Indique un message d'information. Les messages d'information ne stoppent pas l'exécution de mappage.
	Indique un message d'avertissement. Les avertissements ne stoppent pas l'exécution de mappage. Ils peuvent être générés, par exemple, lorsque vous ne créez pas de connexions à des connecteurs d'entrée obligatoires. Dans ces cas, la sortie sera encore générée pour les composants dont les connexions valides existent.
	Indique une erreur. Lorsqu'une erreur se produit, l'exécution de mappage échoue et aucune sortie n'est générée. L'aperçu du code XSLT ou XQuery n'est pas possible non plus.

Pour souligner le composant ou la structure qui a causé l'information, l'avertissement ou le message d'erreur, cliquez sur le texte souligné dans la fenêtre **Messages**.

Actions liées aux messages

La fenêtre **Messages** vous permet de prendre les actions suivantes :

Icône	Description
	Filtrer les messages par sévérité : messages d'information, erreurs, avertissements. Choisir **Tout cocher** pour comprendre tous les niveaux de sécurité (il s'agit du comportement par défaut). Choisir **Tout décocher** pour supprimer tous les niveaux de sévérité du filtre. Dans ce cas, seule l'exécution générale ou le message de statut de validation est affiché.
	Passer à la ligne suivante.
	Passer à la ligne précédente.
	Copier la ligne choisie dans le presse-papiers.
	Copier la ligne choisie dans le presse-papiers, y compris toutes les lignes imbriquées en-dessous.
	Copier les contenus entiers de la fenêtre **Messages** dans le presse-papiers.
	Trouver un texte spécifique dans la fenêtre **Messages**. En option, pour trouver uniquement des mots, choisir **Correspondance mot entier uniquement**. Pour trouver du texte tout en préservant la casse minuscule ou majuscule, choisir **Respecter la casse**.
	Trouver un texte spécifique en commençant depuis la ligne sélectionnée jusqu'à la fin.
	Trouver un texte spécifique en commençant depuis la ligne sélectionnée actuellement jusqu'au début.
	Supprimer la fenêtre Messages.

Lorsque vous travaillez avec plusieurs fichiers de mappage simultanément, vous souhaiterez peut-être afficher des messages d'information, d'avertissement ou d'erreur dans des onglets individuels pour chaque mappage. Dans ce cas, cliquez sur les onglets numérotés disponibles à gauche de la fenêtre de **Messages** avant de valider le mappage.

1.3.4 Volets

Cette rubrique donne un aperçu des volets disponibles.

Volet Mappage

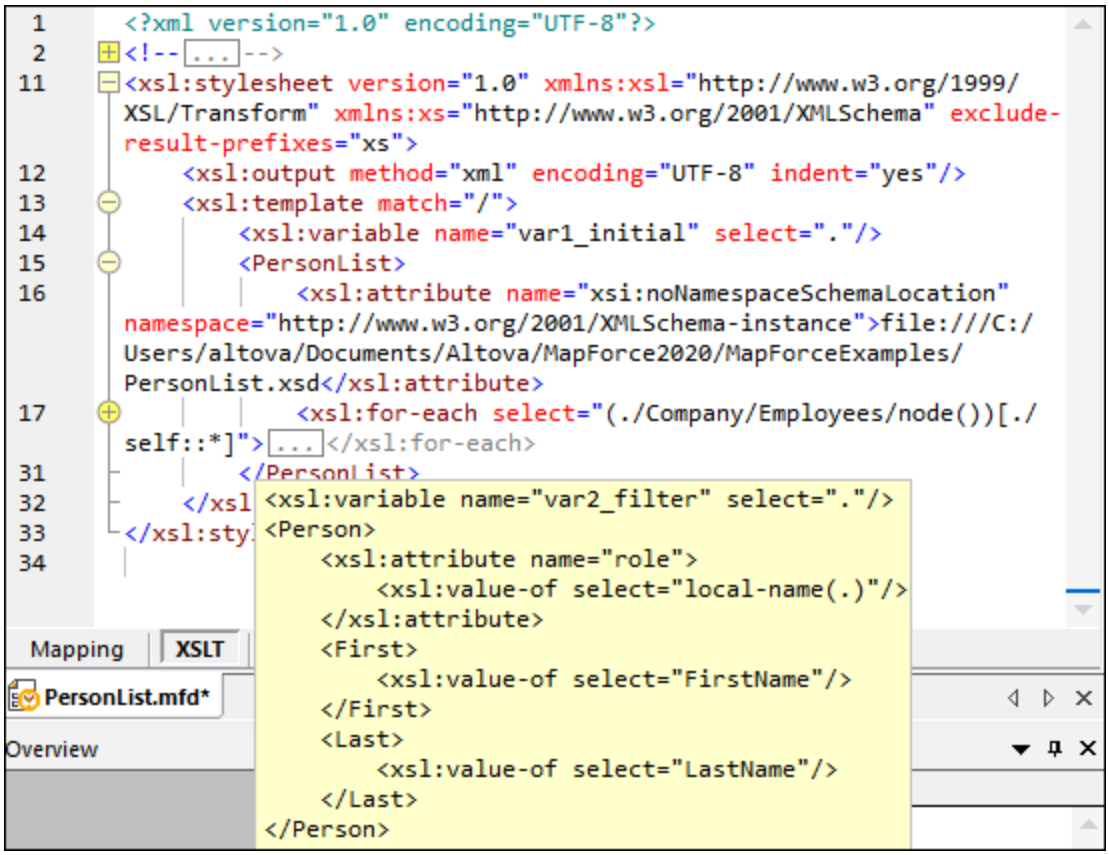
Le volet **Mappage** est la surface de travail dans laquelle vous configurez les mappages 63. Vous pouvez ajouter des composants de mappage (comme des fichiers, des schémas, des constantes, des variables, etc.) dans la zone de mappage à partir du menu **Insérer**. Pour plus d'information, voir Ajouter des composants au

mappage[36]. Vous pouvez aussi glisser des fonctions affichées dans la fenêtre **Bibliothèques** dans le volet **Mappage**. Pour plus d'informations, voir Ajouter une fonction au mappage[195].

Volet XSLT

Le volet **XSLT** affiche le code de transformation XSLT généré depuis votre mappage. Pour passer à ce volet, choisir XSLT, XSLT 2 ou XSLT 3 en tant que langage de transformation[17], puis cliquez sur l'onglet portant le même nom.

Ce volet vous propose une numérotation de ligne et un pliage de code. Pour agrandir ou réduire des portions de code, cliquer sur les icônes "+" et "-" du côté gauche de la fenêtre. Toute portion du code réduit est affichée avec un symbole d'ellipse. Pour consulter le code réduit sans l'agrandir, déplacez le curseur de la souris sur l'ellipse. Une info-bulle s'ouvre, qui affiche le code à consulter comme indiqué dans l'image ci-dessous. Veuillez noter que si le texte consulté est trop grand pour être affiché dans l'infobulle, une ellipse supplémentaire apparaît à la fin de l'infobulle.



Pour configurer les paramètres de vues, y compris l'indentation, les marqueurs de fin de ligne, et autres), cliquez avec la touche de droite sur le volet et choisissez **Paramètres mode texte** du menu contextuel. De manière alternative, cliquez sur (**Paramètres du mode Texte**) dans la barre d'outils.

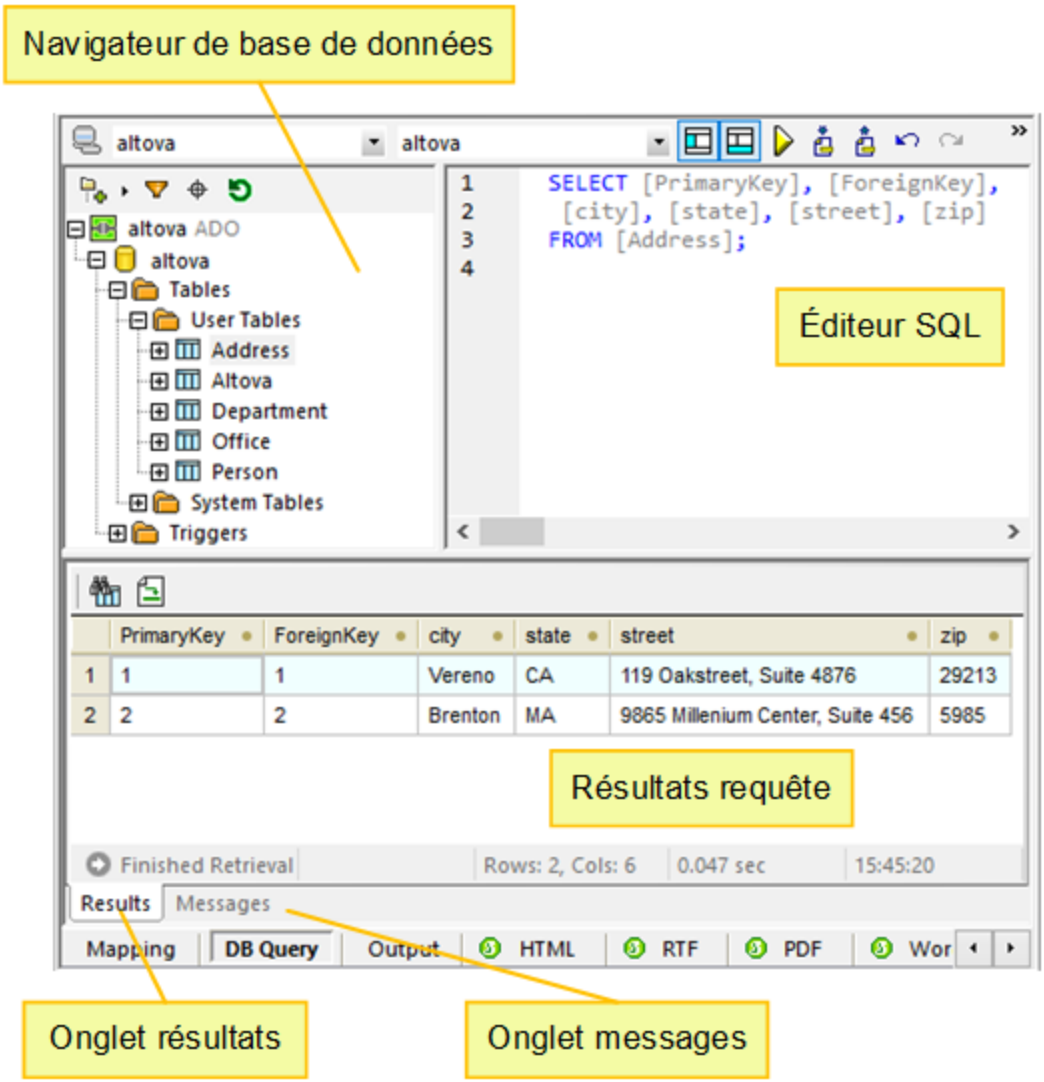
Volet XQuery (Éditions Enterprise et Professional)

Le volet **XQuery** affiche le code de transformation XQuery généré depuis votre mappage, lorsque vous cliquez sur la touche **XQuery**. Ce volet est disponible lorsque vous sélectionnez XQuery en tant que langage de

transformation. Ce volet fournit également des fonctions de numérotation de lignes et de pliage de code, qui fonctionne d'une manière semblable au volet XSLT (voir ci-dessus).

Volet Requête BD (Éditions Enterprise et Professional)

Le volet **Requête BD** vous permet de requêter directement toutes les bases de données principales. Vous pouvez travailler avec plusieurs connexions actives vers des bases de données différentes.



Volet Sortie

Le volet **Sortie** affiche le résultat de la transformation de mappage. Si le mappage génère plusieurs fichiers, vous pouvez naviguer de manière séquentielle à travers chaque fichier généré.

Ce volet fournit également des fonctions de numérotation de lignes et de pliage de code, qui fonctionne d'une manière semblable au volet XSLT (voir ci-dessus).

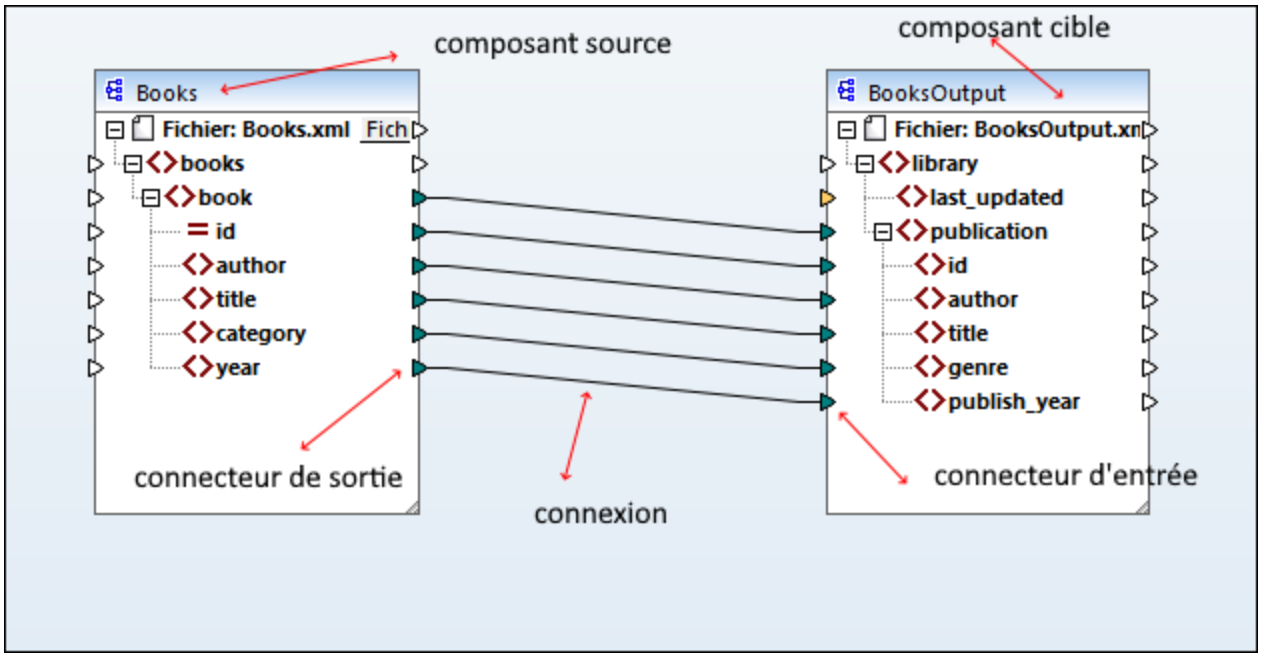
Volets de sortie de StyleVision (Éditions Enterprise and Professional)

Si vous avez installé Altova StyleVision, les volets de sortie de StyleVision deviennent disponibles à côté du volet **Sortie**. Les volets de sortie de StyleVision vous permettent de consulter et d'enregistrer la sortie de mappage dans des formats HTML, RTF, PDF et Word 2007+. Cela est possible grâce aux fichiers StyleVision Power Stylesheet (SPS) conçus dans StyleVision et attribués à un composant de mappage dans MapForce.

2 Notions fondamentales de mappage

Un design de mappage MapForce (ou simplement un "mappage") est la représentation visuelle de la manière dont les données sont transformées d'un format à un autre. Un mappage consiste en des *composants* que vous ajoutez à la zone de mappage MapForce afin de créer vos transformations de données. Un mappage valide consiste en un ou plusieurs *composants de source* connectés à un ou plusieurs *composants cibles*. Vous pouvez exécuter un mappage et consulter son résultat directement dans MapForce. Vous pouvez générer du code et l'exécuter extérieurement. Vous pouvez aussi compiler un mappage dans un fichier d'exécution MapForce et automatiser l'exécution de mappage MapForce Server ou FlowForce Server. MapForce enregistre les mappages comme fichiers `.mfd`.

La capture d'écran ci-dessous illustre la structure basique d'un mappage de :



Nouveau mappage

Pour créer un nouveau mappage, cliquez sur (**Nouveau**) dans la barre d'outils. En alternative, cliquez sur **Nouveau** dans le menu **Fichier**. La prochaine étape consistera à ajouter des composants 36 au mappage et à créer les connexions 46.

Parties principales d'un mappage

Les sous-sections ci-dessous décrivent les parties principales d'un design de mappage.

Composant

Dans MapForce, le terme *composant* est ce qui représente visuellement la structure de vos données, ou indique la manière dont les données doivent être transformées. Les composants sont les pièces centrales de tout mappage et sont représentés en tant que boîtes rectangulaires. Les composants peuvent être divisés en deux grands groupes :

- Les composants source et cible

- Les composants de structure[111] et de transformation[146]

Veuillez noter que ces deux groupes ne sont pas mutuellement exclusifs. Le premier groupe reflète les relations entre les composants ; par ex., un composant peut être la source pour un composant et la cible pour un autre composant. MapForce lit les données depuis un composant source et écrit ces données dans un composant cible. Lorsque vous exécutez un mappage, le composant cible instruit MapForce soit à générer un fichier (ou plusieurs fichiers), soit à sortir le résultat en tant que valeur de string pour un traitement ultérieur dans un programme externe. Les types de composants du premier groupe sont décrits ci-dessous :

- Une*source* est située à gauche du composant cible. MapForce lit les données depuis la source.
- Une *cible* est située à droite de la source. MapForce écrit les données dans le composant cible.
- Un composant *pass-through* est un sous-type de composants source et cible. Un composant pass-through agit en tant que source et cible. Pour plus d'information, voir les Mappages en chaîne[94]. Veuillez noter que seuls les composants de structure peuvent être pass-through.

Le deuxième groupe (composants structurels/de transformation) affiche si un composant est doté d'une structure de données ou s'il est utilisé pour transformer les données mappées d'un autre composant.

Pour en savoir plus sur les composants et actions liées au composant, voir Components[32].

Connecteur
Un connecteur est un petit triangle affiché sur le côté gauche ou droite d'un composant. Les connecteurs d'entrée se trouvent à gauche d'un composant et affichent des points d'entrée de données *à ce composant*. Les connecteurs de sortie se trouvent à droite d'un composant et affichent des points de sortie de données *de ce composant*.

Connexion
Une connexion est une ligne que vous pouvez tirer entre deux connecteurs. En créant des connexions, vous instruisez MapForce de transformer des données d'une manière spécifique : par exemple, lire des données depuis un document XML et l'écrire dans un autre document XML.

Dans cette section

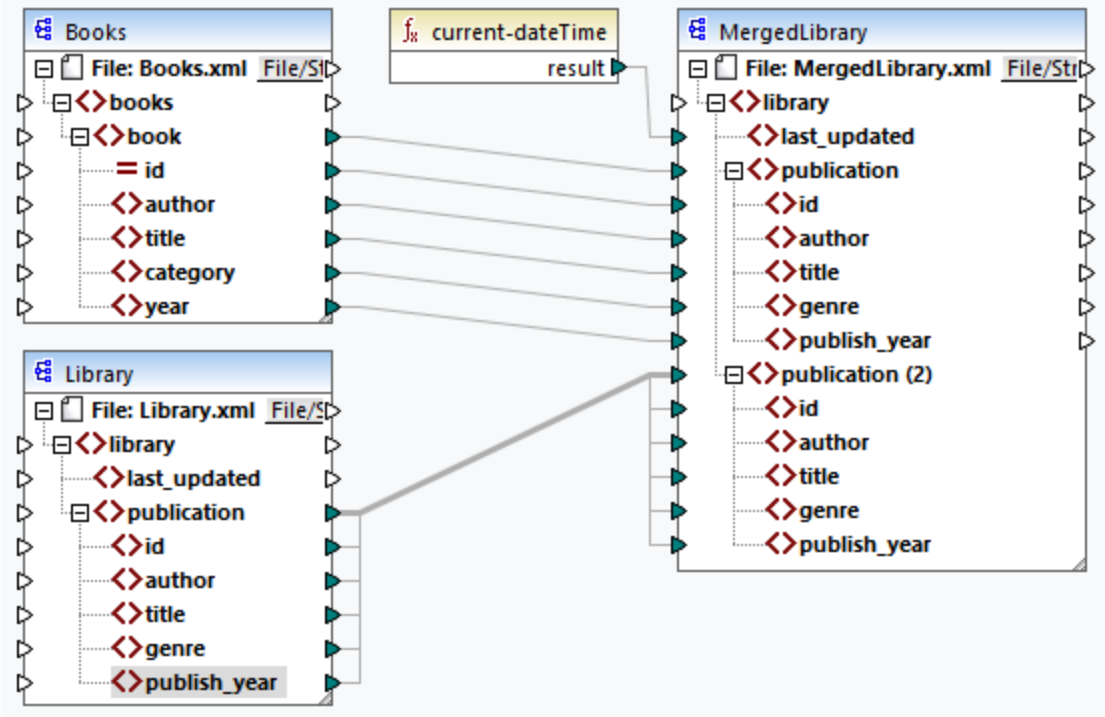
Cette section décrit les tâches et concepts les plus communs de MapForce. La section est organisée en sous-sections comme suit :

- Composants[32]
- Connexions[46]
- Procédures générales et fonctions[63]

2.1 Composants

Les composants sont des éléments centraux de tout design de mappage dans MapForce. Visuellement, les composants sont représentés comme boîtes rectangulaires dans la zone de mappage. Cette rubrique donne un aperçu des composants structurels et de transformation (*voir exemple ci-dessous*). La distinction est basée sur le fait si un composant a une structure de données ou s'il est utilisé pour transformer des données. Voir la description de ces deux types dans les sous-sections ci-dessous. Voir aussi Notions fondamentales de mappage 30. Outre les composants de structure et de transformation, vous pouvez ajouter des commentaires à votre mappage (*voir Commentaires ci-dessous*).

Exemple de composants
L'exemple de mappage ci-dessous illustre deux composants de source de données (`Books` et `Library`), un composant cible de données (`MergedLibrary`), et un composant de transformation (la fonction **`current-dateTime`**).



Composants de structure

Les composants structurels représentent une structure abstraite de vos données (par ex., un fichier XML). La liste des composants structurels qui peuvent être utilisés dans les sources et cibles de données se trouve sous Composants structurels 111. Les composants structurels peuvent lire les données depuis quelque/s source/s, écrire des données dans certaine(s cible/s, ou encore stocker des données à des étapes intermédiaires dans le processus de mappage (par ex., afin de prévisualiser des données). La table ci-dessous donne un aperçu des composants de structure et de leurs boutons de barre d'outils respectifs.

Icône	Description
	Composant XML
	Composant de texte (*éditions Professional et Enterprise*)
	Composant de base de données (*éditions Professional et Enterprise* pour les bases de données SQL ; *Enterprise Edition* pour les bases de données NoSQL)
	Composant JSON (*Enterprise Edition*)
	Composant Microsoft Excel (*Enterprise Edition*)
	Composant EDI (*Enterprise Edition*)
	Composant XBRL (*Enterprise Edition*)
	Protocol Buffers (*Enterprise Edition*)

Composants de transformation

Les composants de transformation vous aide à transformer les données[194], stocker un résultat de mappage intermédiaire[158] pour un traitement ultérieur, remplacer une valeur par une autre valeur[182], trier[170], grouper[276], et filtrer[176] vos données. La table ci-dessous donne un aperçu des composants de transformation et de leurs boutons de barre d'outils respectifs.

Icône	Description
	Entrée simple
	Sortie simple
	Composant de filtre
	Composant de tri
	Fonction intégrée
	Fonction définie par l'utilisateur
	Composant SQL/NoSQL-WHERE/ORDER (*éditions Professional et Enterprise*)
	Composant Value- Map
	Variable
	Fonction de service Web (*Enterprise Edition*)
	Exception (*éditions Professional et Enterprise*)

Icône	Description
	Constante
	Condition If-Else
	Composant Join (*éditions Professional et Enterprise*)

Commentaires

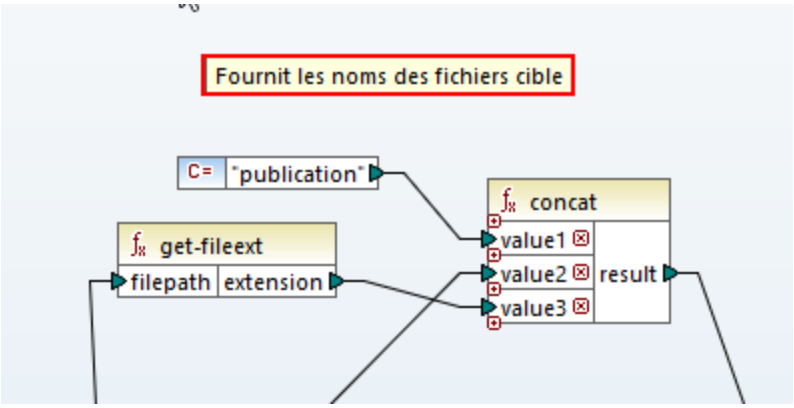
MapForce permet d'ajouter des commentaires en composants standalone et comme notes sous les composants existants.

Composants commentaires

Les composants commentaires sont des boîtes autoportantes et ne peuvent pas être connectés avec tout autre composant. Pour ajouter un composant commentaires, vous pouvez sélectionner une des options suivantes :

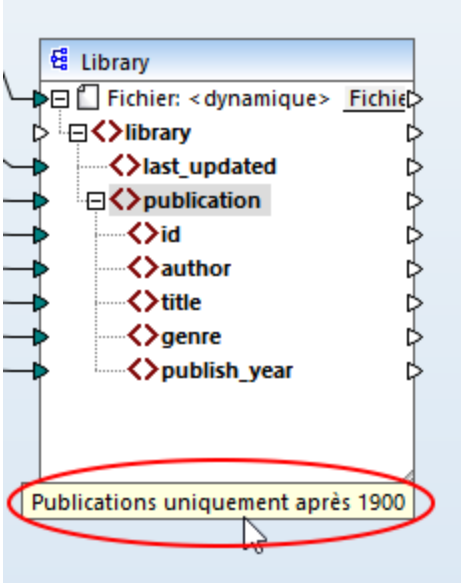
- Sélectionnez la commande de barre d'outils  , qui ouvre un dialogue où vous pouvez saisir votre commentaire.
- Sélectionnez la commande de menu **Insérer | Commentaire**, qui ouvre un dialogue où vous pouvez saisir votre commentaire.
- Double-cliquez la zone vide de votre mappage, saisissez le symbole #, tapez un commentaire et appuyez sur **Enter**. Le symbole # n'apparaîtra pas dans la zone de commande.

Pour déplacer une barre de commentaires, glissez-la à l'emplacement souhaité. Pour supprimer un commentaire, cliquez sur la zone de commentaires et appuyez sur la touche **Supprimer**. Un exemple de composant de commentaire est illustré ci-dessous (*boîte rectangulaire rouge*).



Composants Commentaires

Outre les boîtes de commentaire autoportant, vous pouvez aussi ajouter des commentaires aux composants existants. De tels commentaires sont affichés en-dessous du composant (*cercle rouge ci-dessous*).



Pour ajouter un commentaire sous un composant existant, vous pouvez choisir une des options suivantes :

- Cliquez avec la touche de droite à l'intérieur du composant et sélectionnez **Éditer Commentaire** depuis le menu contextuel. Ceci ouvre le dialogue où vous pouvez saisir votre commentaire.
- Sélectionnez un composant auquel vous voulez ajouter un commentaire. Puis, sélectionnez **Éditer Commentaire** dans le menu **Composant**. Ceci ouvre le dialogue où vous pouvez saisir votre commentaire.

L'affichage des commentaires de composant peut être limité á un nombre spécifique de lignes, qui peuvent être définies dans le menu **Outils | Options | Général | Aperçu mappage**. Pour plus d'informations, voir Options 469.

Pour supprimer un commentaire de composant, suivez une des étapes suivantes :

- Double-cliquez sur le commentaire, supprimer tout le texte et cliquez sur **Enter**.
- Cliquez avec la touche de droite sur le commentaire ou à l'intérieur du composant, sélectionnez **Éditer Commentaire** depuis le menu contextuel, supprimer le texte et cliquez sur **OK**.

Éditer les Commentaires
Vous pouvez éditer les deux types de commentaires d'une des manières suivantes :

- Double-cliquez sur le texte du commentaire et lancez l'édition directement dans la boîte. Puis, appuyez sur **Enter**.
- Cliquez avec la touche de droite dans la zone de commande, éditez du texte dans le dialogue **Éditer commentaire** et cliquez sur **OK**. Pour les commentaires de composant, vous pouvez aussi accéder le dialogue **Éditer Commentaire** en cliquant avec la touche de droite sur celui-ci et sélectionner l'option **Éditer Commentaire** dans le menu contextuel.

Dans cette section

Cette section donne un aperçu des composants et est organisée comme suit :

2.1.1 Ajouter des composants au mappage

Cette rubrique explique comment ajouter des composants au mappage. Pour ajouter un composant, vous allez d'abord devoir créer un nouveau design de mappage 30, ensuite procédez comme suit :

- Dans le menu **Insérer**, choisissez un type de composant (par ex., **XML Schéma/Fichier**).
- Glissez un fichier depuis Windows File Explorer dans la zone de mappage.
- Cliquez sur le bouton pertinent dans la barre d'outils **Insérer Composant** (*voir la capture d'écran ci-dessous*)

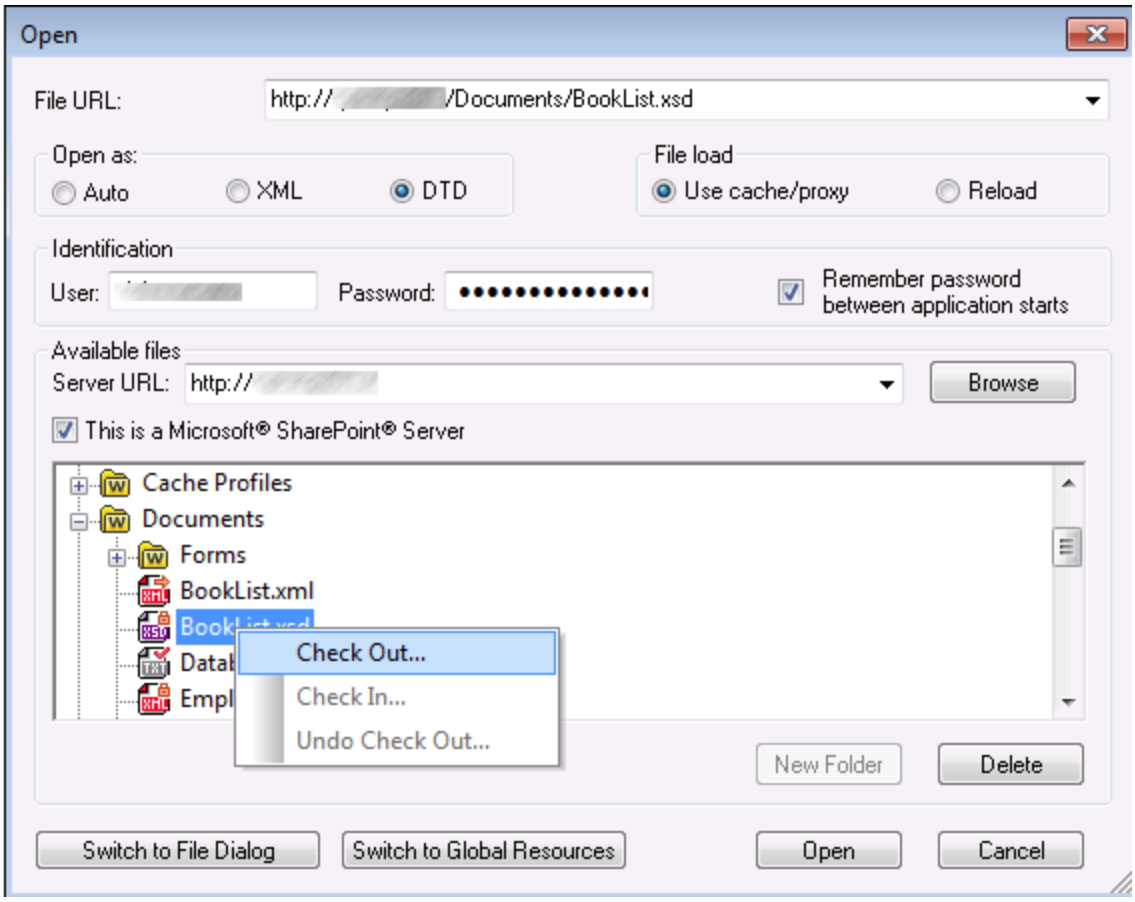
Chaque type de composant a un but et une fonction spécifiques. Pour obtenir un aperçu des composants, voir Composants 32. Si vous souhaitez en savoir plus sur les structures de données qui peuvent être utilisées comme sources et cibles de, voir les composants de structure 111. Pour des informations concernant les composants intégrés de MapForce utilisés pour stocker des données temporairement ou les transformer, voir Composants de transformation 146.

Lorsque vous voulez ajouter un composant structurel á votre mappage, vous pouvez choisir d'ajouter un fichier local, un composant d'un URL ou d'une liste de ressources globales (*voir les sous-sections ci-dessous*).

Ajouter des composants depuis l'URL

Ajouter des composants depuis un URL est pris en charge uniquement pour les composants de source 30. Les protocoles pris en charge sont HTTP, HTTPS et FTP. Dépendant du type de la structure de données, les instructions relatives à l'ajout d'un composant depuis un URL peuvent varier. Pour la plupart des structures de données, appliquez les instructions suivantes :

1. Sélectionnez le type de composant que vous souhaitez ajouter (par ex., **XML Schéma/Fichier**).
2. Cliquez sur **Passer à l'URL** dans la boîte de dialogue **Ouvrir**.
3. Saisissez l'URL du fichier dans le champ de saisie *Fichier URL* et cliquez sur **Ouvrir** (*voir ci-dessous*).



La liste ci-dessous décrit les options disponibles dans le dialogue **Ouvrir**.

- *Ouvrir comme :* Cette option définit la grammaire pour le parser. L'option par défaut et recommandée est *Auto*.

- *Chargement du fichier :* Si le fichier que vous chargez ne changera probablement pas, sélectionnez l'option *Utiliser cache/proxy* pour mettre en cache les données et accélérer le chargement du fichier. Sélectionnez *Recharger* si vous voulez recharger le fichier à chaque fois que vous ouvrez le mappage.

- *Identification :* Si le serveur nécessite une authentification de mot de passe, vous serez invité à saisir le nom d'utilisateur et le mot de passe. Si vous voulez que votre nom utilisateur et mot de passe soient enregistrés la prochaine fois, sélectionnez la case à cocher *Se rappeler le mot de passe quand l'application démarre*.

- *Serveur URL :* Pour des serveur qui prennent en charge Web Distributed Authoring et Versioning (WebDAV), vous pouvez chercher des fichiers une fois après avoir saisi l'URL de serveur dans le champ de saisie *URL Serveur* et cliquez sur **Parcourir**. Bien que l'aperçu montre tous les types de fichier, assurez-vous d'ouvrir le type de fichier comme à l'étape 1 ci-dessus. Autrement, des erreurs surviendront.

- *Check in/out :* Si le serveur est un Microsoft SharePoint Server, sélectionnez la case à cocher *Il s'agit d'un serveur Microsoft® SharePoint®*. Cela affichera état check-in/ check-out du fichier dans la zone d'aperçu. Si vous souhaitez vous assurer que personne ne puisse éditer le fichier sur le serveur

pendant que vous l'utilisez, cliquez avec la touche de droite sur le fichier et sélectionnez **Check Out** (*voir la capture d'écran ci-dessous*). Pour faire le check-in de tout fichier préalablement en « checked-out », cliquez avec la touche de droite sur le fichier et sélectionnez **Check In**.

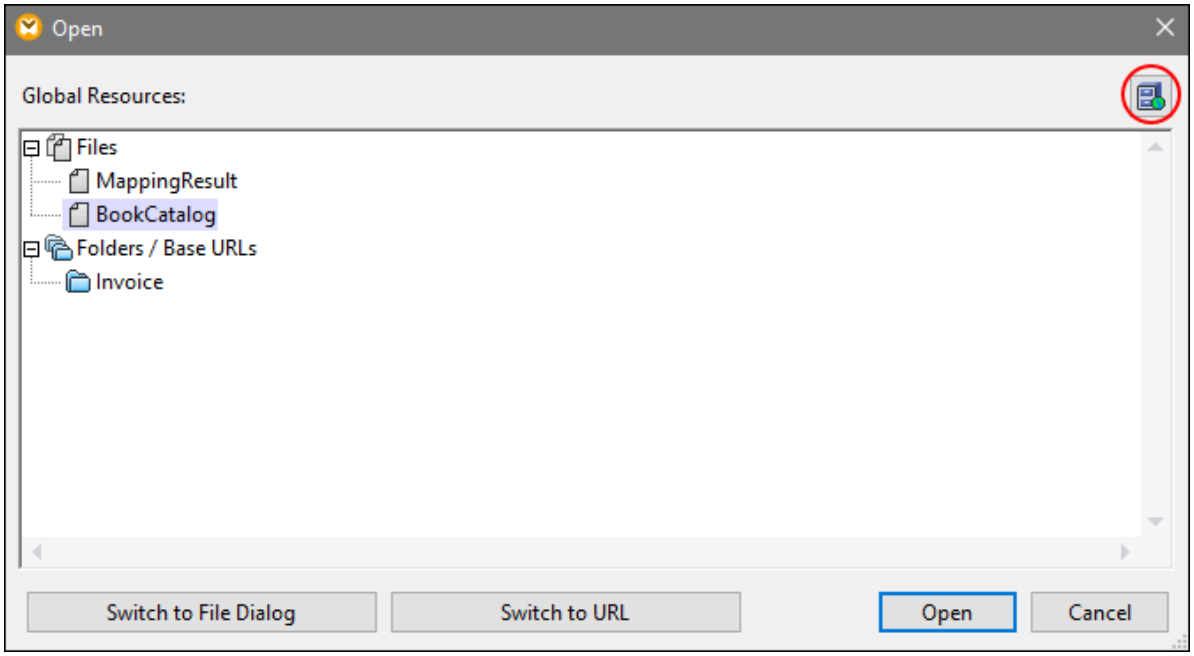
- *Passer au dialogue Fichier :* En cliquant sur le bouton, vous serez redirigé vers le dialogue dans lequl vous pouvez sélectionner un fichier local.

- *Commutateur vers les Ressources globales :* En cliquant sur le bouton, vous serez redirigé vers le dialogue qui autorise de sélectionner une ressource globale.

Ajouter les ressources globales

Si vous avez défini une base de données (*éditions Professional et Enterprise*), un fichier ou un dossier comme ressource globale, vous pouvez l'ajouter à votre mappage. Pour plus d'informations sur les Ressources globales, voir Ressources globales Altova[430]. Dépendant du type de structure de données avec laquelle vous travailler, les instructions relatives à l'ajout d'une ressource globale peuvent varier. Pour la plupart des structures de données, appliquez les instructions suivantes :

1. Sélectionnez le type de composant que vous souhaitez ajouter (par ex., **XML Schéma/Fichier**).
2. Cliquez sur **Ressources globales** dans la boîte de dialogue **Ouvrir**.
3. Sélectionnez une des ressources de la liste et cliquez sur **Ouvrir** (*voir ci-dessous*).



Si vous voulez ajouter, éditer ou supprimer des ressources globales, cliquez sur l'icône **Gérer Ressources globales** (*encerclé en rouge ci-dessus*). Le dialogue **Ouvrir** pour les ressources globales vous permet de retourner au dialogue avec des fichiers locaux (**Passer au Dialogue Fichier**) ou ouvrir un fichier depuis une URL (**Basculer vers l'URL**).

2.1.2 Les bases de composant

Cette rubrique explique comment définir, chercher et manipuler les composants de structure[32]. Pour plus d'information, veuillez voir les sous-sections ci-dessous.

Changer les paramètres de composant

Après avoir ajouté un composant dans la zone de mappage, vous pouvez configurer son paramètres applicables depuis le dialogue **Paramètres de composant**. Vous pouvez ouvrir le dialogue **Paramètres de composant** d'une des manières suivantes :

- Double-cliquez sur l'en-tête de composant.
- Sélectionner le composant et cliquez sur les **Propriétés** dans le menu **Composant**.
- Cliquez avec la touche de droite sur l'en-tête du composant, cliquez sur **Propriétés**.

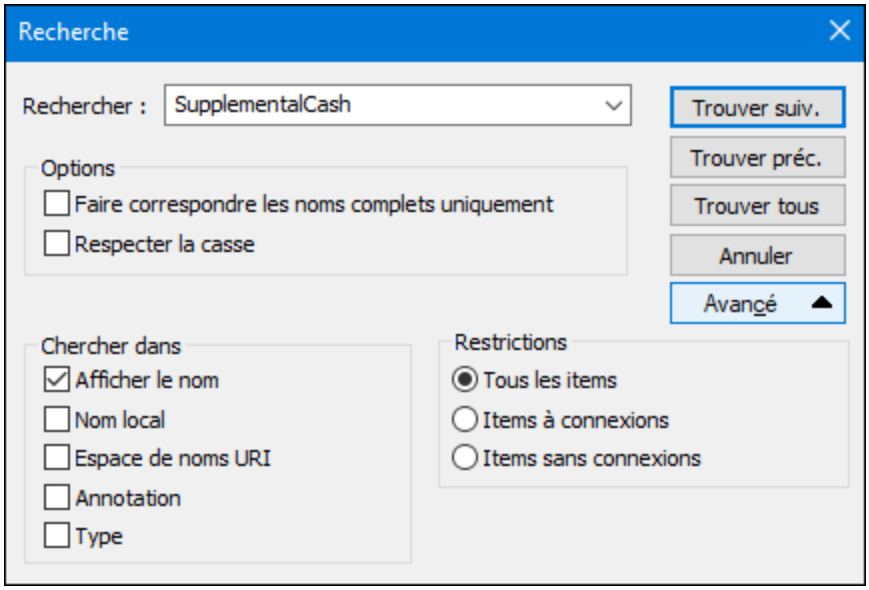
Voir la liste de paramètres disponibles dans Paramètres de composant XML[113].

Pour chaque composant basé sur fichier (par ex., un fichier XML) la touche File (*Édition de base*) ou File/String (*éditions Professional et Enterprise*) apparaît à côté du nœud racine. Ce bouton spécifie des options avancées pour traiter ou générer de multiples fichiers en un seul mappage. Pour plus d'informations, voir Traiter plusieurs fichiers d'entrée ou de sortie[400].

Chercher un composant

Pour rechercher un nœud spécifique dans un composant, suivez les étapes suivantes :

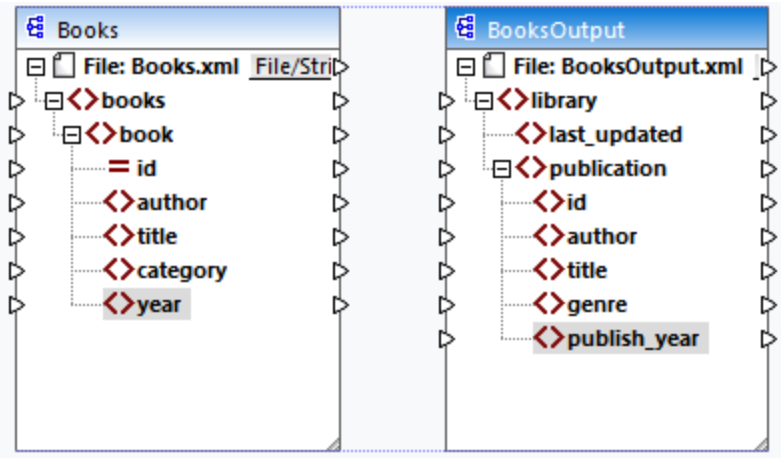
1. Cliquez sur le composant que vous souhaitez parcourir et appuyer sur les touches **CTRL+F**.
2. Saisissez un terme de recherche et cliquez sur **Trouver suivant/précédent/tout** (*voir la capture d'écran ci-dessous*).



Utiliser les options **Avancé** pour définir quel(s) item(s) (nœuds) doivent être recherchés. Vous pouvez aussi restreindre les options de recherche basées sur les connexions spécifiques.

Aligner des composants

Lorsque vous déplacez des composants dans la zone de mappage, MapForce affiche des repères (lignes pointillées) qui vous aident à aligner des composants (*voir la capture d'écran ci-dessous*).



Pour activer cette option, suivez les étapes ci-dessous :

1. Allez au menu **Outils** et cliquez sur **Options**.
2. Dans le groupe **Édition**, cochez la case à cocher **Aligner les composants sur glissage de souris**.

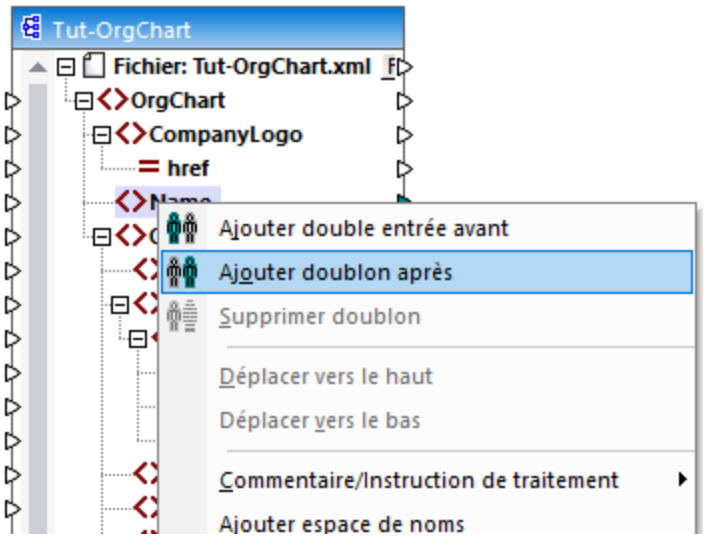
Dupliquer l'entrée

Parfois, vous devez configurer un composant pour qu'il puisse accepter les données provenant de plusieurs sources. Si vous voulez que le schéma cible accepte les données de plus d'un schéma source, vous pouvez dupliquer tout nœud d'entrée dans votre composant cible. La duplication d'entrées n'a de sens que pour un composant cible : Les nœuds dupliqués ne peuvent accepter que des données, mais il n'est pas possible de mapper des données depuis des nœuds dupliqués. Vous pouvez dupliquer autant de nœuds que vous souhaitez.

Il existe deux manières de dupliquer l'entrée : (i) en sélectionnant **Ajouter double entrée Avant/ Après** depuis le menu contextuel et (ii) en connectant un nœud source avec un nœud cible, qui est déjà connecté à un nœud différent. La première option est décrite ci-dessous. Des informations sur la deuxième option peuvent être trouvées dans le deuxième tutoriel 89.

Ajouter double entrée Avant/Après

Pour dupliquer un nœud d'entrée particulier, cliquez dessus avec la touche de droite et sélectionnez **Ajouter double entrée Avant/ Après** depuis le menu contextuel (*voir la capture d'écran ci-dessous*). Dans l'image ci-dessous, le nœud `author` est en train d'être dupliqué de manière à ce que les données puissent être mappées vers le nœud dupliqué d'une autre source de nœud.



Note : la duplication d'attributs XML n'est pas permise, puisqu'elle rendra l'instance XML invalide.

2.1.3 Chemins de fichier

Un fichier de design de mappage (**`*.mfd`**) peut contenir des références à plusieurs schémas et fichiers d'instance. MapForce utilise les fichiers de schéma afin de déterminer la structure des données à mapper. Dans les éditions MapForce Professional et Enterprise, les mappages peuvent aussi contenir des références aux fichiers StyleVision Power Stylesheets (**`*.sps`**), utilisés pour formater des données pour les sorties comme PDF, HTML et Word. Les mappages peuvent aussi contenir des références à des bases de données basées sur des fichiers comme Microsoft Access ou SQLite.

Les références à des fichiers sont créées par MapForce lorsque vous ajoutez un composant au mappage. Néanmoins, vous pouvez toujours définir ou changer les références de chemin manuellement, le cas échéant.

Cette section fournit des instructions sur comment définir ou changer les chemins vers différents types de fichier référencés par un mappage. La section est organisée en rubriques suivantes :

2.1.3.1 Chemins relatifs et absolus

Cette rubrique explique comment utiliser les chemins absolus et relatifs des fichiers référencés par votre composant. Un chemin absolu affiche tout l'emplacement d'un fichier, en commençant par un répertoire racine (*voir Exemple :. Paramètres de composant XML*). Un chemin relatif affiche l'emplacement du fichier qui est relatif au répertoire de travail actuel : par ex., **`Books.xml`**.

Dans la boîte de dialogue des **Paramètres de composant** (*voir l'exemple ci-dessous*), vous pouvez spécifier les chemins absolus ou relatifs pour de nombreux fichiers référencés par le composant. La liste des fonctions d'expression est indiquée ci-dessous :

- Fichiers d'entrée à partir desquels MapForce lit des données ;
- Fichiers de sortie vers lesquels MapForce écrit des données ;
- Fichiers de schéma qui s'appliquent aux composants avec un schéma ;
- Fichiers de structure utilisés par des paramètres d'entrée ou de sortie des fonctions définies par l'utilisateur et des variables ;
- Fichiers StyleVision Power Stylesheet (**`*.sps`**) utilisés pour formater des données pour les sorties comme les PDF, HTML et Word.
- Les fichiers de base de données dans le cas de composants de base de données (*éditions Professional et Enterprise*).

Copier-coller et chemins relatifs
Lorsque vous copiez un composant depuis un mappage et que vous le collez dans un autre mappage, MapForce vérifie si les chemins relatifs de fichiers de schéma peuvent être résolus par rapport au dossier du mappage de destination. Si le chemin ne peut pas être résolu, vous serez invité à rendre les chemins relatifs en absolu.

Chemins cassés
Lorsque vous ajoutez ou modifiez une référence de fichier dans un mappage, et que le chemin ne peut pas être résolu, MapForce affiche un message d'avertissement. Les références de chemin cassé peuvent arriver dans les cas suivants :

- Vous utilisez des chemins relatifs, puis déplacez le fichier de mappage vers un nouveau répertoire sans déplacer les fichiers de schéma et d'instance.
- Vous utilisez des chemins absolus vers des fichiers dans le même répertoire que le fichier de mappage, puis déplacez le répertoire vers un autre emplacement.

Lorsqu'un de ces scénarios arrivent, MapForce marque le composant en rouge. La solution dans ce cas est de double-cliquer sur l'en-tête de composant et de mettre à jour toutes les références de chemin cassés dans le dialogue **Paramètres de composant**. Voir aussi Changer les paramètres de composant 39.

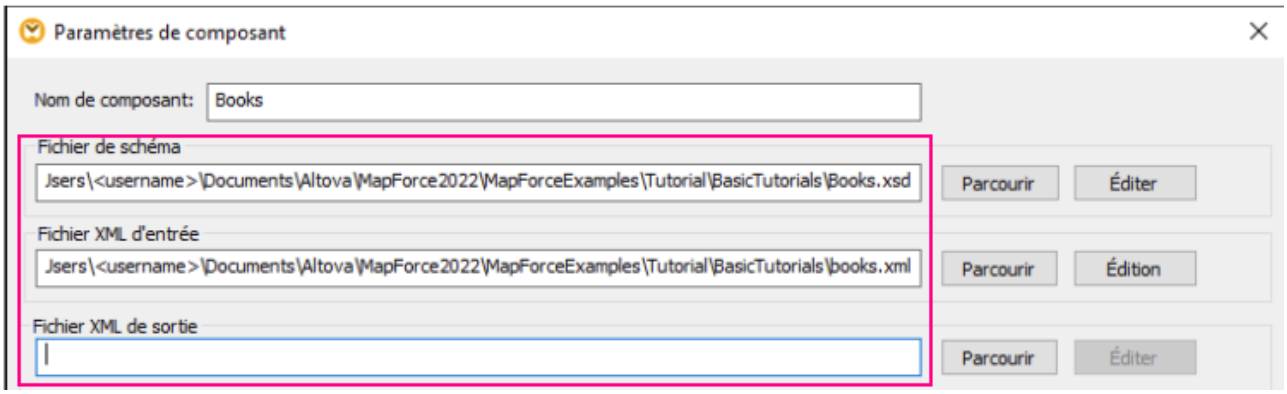
Exemple : Composant XML

L'exemple ci-dessous affiche comment les chemins de fichier sont utilisés dans un composant XML. Si vous souhaitez enregistrer tous les fichiers relatifs au mappage dans le fichier de mappage (**`.mfd`**), vérifiez la boîte *Enregistrer tous les chemins de fichier relatifs au fichier MFD* dans le bas de la boîte de dialogue **Paramètres de composant**. Il s'agit d'une option recommandée et par défaut qui affecte tous les fichiers référencés par le composant (*affiché dans le cadre rouge dans l'image ci-dessous*). Si vous n'avez pas encore enregistré votre mappage, vous verrez des chemins absolus au schéma et/ou des fichiers d'instance dans la boîte de dialogue **Paramètres de composant**. Pour voir les chemins relatifs dans la boîte de dialogue **Paramètres de composant**, suivez les étapes suivantes :

1. Créer un nouveau mappage 30 et ajouter un composant de structure 36 : par ex., un fichier XML avec un schéma XML y assigné.
2. Double-cliquez sur l'en-tête du composant pour ouvrir une boîte de dialogue **Paramètres de composant**.
3. Vérifier la boîte *Enregistrer tous les chemins de fichier relatifs au fichier MFD* en bas de la boîte de dialogue **Paramètres de composant**.
4. Enregistrer votre mappage.

5. Vous pouvez ouvrir maintenant une nouvelle fois les **Paramètres de composant** afin de vérifier les chemins relatifs dans les champs de texte pertinents.

Note : les chemins qui référencent un pilote non-local ou utilisent une URL ne seront par rendus relatifs.



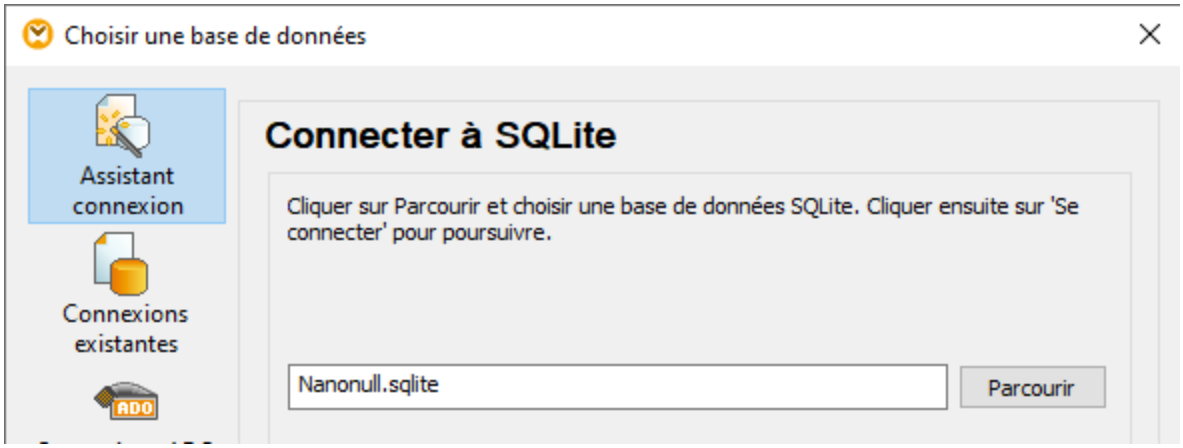
Quand la case à cocher *Enregistrer tous les chemins de fichier relatifs au fichier MFD* est sélectionnée, MapForce assurera le suivi des fichiers référencés par le composant même si vous enregistrez le mappage dans un nouveau dossier. Si tous les fichiers se trouvent dans le même répertoire que le mappage, les références de chemin ne seront pas brisées lorsque vous déplacez le répertoire complet vers un nouvel emplacement sur le disque.

Le paramètre *Enregistrer tous les chemins de fichier relatifs au fichier MFD* s'applique aux fichiers suivants :

- Fichiers de structure utilisés par des paramètres complexes d'entrée ou de sortie des fonctions définies par l'utilisateur et des variables de type complexe ;
- Fichiers plats d'entrée ou de sortie (*éditions Professional et Enterprise*) ;
- Fichiers de schéma référencés par les composants de base de données qui prennent en charge des champs XML (*éditions Professional et Enterprise*) ;
- Fichiers de base de données (*éditions Professional et Enterprise*) ;
- XBRL d'entrée ou de sortie, FlexText, EDI, Excel 2007+, fichiers JSON (*uniquement Enterprise Edition*).

Exemple : Composant de base de données (éditions Professional et Enterprise)

Quand vous ajoutez un fichier de base de données tel que Microsoft Access ou SQLite au mappage, vous pouvez saisir un chemin relatif à la place d'un chemin absolu dans la boîte de dialogue **Sélectionner une base de données** (*voir la capture d'écran ci-dessous*). Avant de saisir des chemins de fichier relatifs, assurez-vous d'enregistrer d'abord le fichier de mappage **`.mfd`**. Si vous voulez changer le chemin d'un composant de données qui est déjà dans le mappage, cliquez sur **Changer** dans la boîte de dialogue **Paramètres de composant**.



Note : lorsque vous générez un code de programme, compilez des fichiers d'exécution MapForce Server (**.mfx**), ou déployez le mappage sur FlowForce Server, un chemin relatif sera converti dans un chemin absolu si vous sélectionnez la case à cocher *Rendre les chemins absolus dans le code généré* dans les paramètres de mappage 74. Pour en savoir plus, voir À propos des chemins dans le code généré 44.

2.1.3.2 Les chemins dans des environnement d'exécution différents

Si vous générez du code depuis les mappages, les fichiers générés sont exécutés dans l'environnement cible de votre choix : par exemple, RaptorXML Server. Pour que le mappage puisse être exécuté avec succès, tout chemin relatif doit être explicite dans l'environnement dans lequel le mappage est exécuté. Les chemins de base pour chaque langage cible sont indiqués ci-dessous :

Langage cible	Chemin de base
XSLT, XSLT2, XSLT3	Chemin du fichier XSLT.
XQuery*	Chemin du fichier XQuery.
C++, C#, Java*	Répertoire de travailde l'application générée.
Built-in* (pour les aperçus du mappage dans MapForce)	Chemin du fichier de mappage (**.mfd**).
Built-in* (pour l'exécution du mappage avec MapForce Server)	Répertoire de travail actuel.
Built-in* (pour l'exécution du mappage avec MapForce Server sous le contrôle de FlowForce Server)	Répertoire de travail de la tâche ou répertoire de travail de FlowForce Server.

** Langages disponibles dans MapForce Professional et Enterprise editions*

Chemin relatif vers chemin absolu
Lorsque vous générez un code de programme, compilez des fichiers d'exécution MapForce Server (**.mfx**), ou déployez le mappage sur FlowForce Server, un chemin relatif sera converti dans un chemin absolu si vous

sélectionnez la case à cocher **Rendre les chemins absolus dans le code généré** dans les paramètres de mappage[74].

Lorsque vous générez un code et que la case à cocher a été sélectionnée, MapForce résout tout chemin relatif basés sur le répertoire du fichier de mappage **.mfd**, et les rend absolus dans le code généré. Ce paramètre a une incidence sur les fichiers suivants :

- Fichiers d'instance d'entrée et de sortie pour des composants basés sur le fichier ;
- Fichiers Access et base de données SQLite utilisés comme composants de mappage (*éditions Professional et Enterprise*).

Chemins de bibliothèque dans du code généré

Les fichiers de mappage peuvent contenir en option des références de chemin à des bibliothèques de types variés. Par exemple, vous pouvez importer des fonctions définies par l'utilisateur depuis un autre fichier de mappage, depuis XSLT personnalisé, XQuery*, C#* ou des bibliothèques Java*, ou depuis des fichiers **.mff*** (fonction de MapForce). Pour plus d'informations, voir Gérer les Bibliothèques de fonction[198].

* *Fonctions disponibles dans les éditions MapForce Professional et Enterprise*

L'option **Rendre les chemins absolus dans le code généré** ne s'applique qu'aux composants de mappage, et elle ne touche pas les chemins menant aux bibliothèques externes. Pour toutes les bibliothèques différentes de XSLT et XQuery, le chemin de bibliothèque sera converti en un chemin absolu dans le code généré. Par exemple, si votre fichier de mappage contient des références de bibliothèque comme des fichiers .NET **.dll** ou Java **.class**, et que vous souhaitez exécuter le code généré dans un autre environnement, les bibliothèques référencées doivent exister dans le même chemin dans l'environnement cible.

Si vous prévoyez de générer un fichier XSLT ou XQuery depuis un mappage, vous pouvez rendre le chemin de bibliothèque relatif au fichier XSLT ou XQuery généré, comme suit :

1. Ouvrez les paramètres de mappage[74].
2. Sélectionnez la case à cocher **Bibliothèques de référence avec des chemins relatifs aux fichiers XSLT/XQuery générés.**. Assurez-vous que la bibliothèque XSLT ou XQuery existe dans ce chemin.

2.2 Connexions

Une connexion est une ligne qui connecte une source vers une cible. Les connexions représentent visuellement comment les données sont mappées d'un nœud vers un autre. Les sous-sections ci-dessous décrivent différentes actions liées à la connexion que vous pouvez effectuer.

Créer, copier, supprimer une connexion

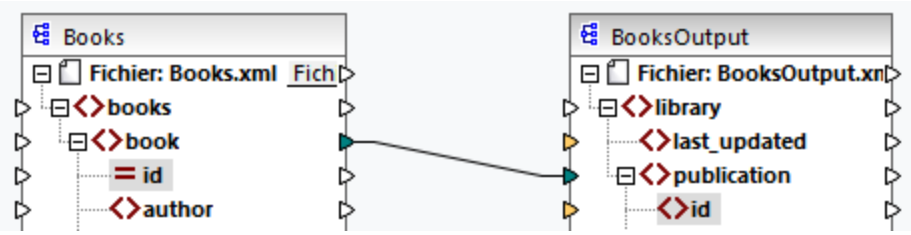
Pour créer une connexion entre deux lignes, appuyez et restez sur le connecteur de sortie[31] d'un nœud source et glissez-le vers un nœud de destination. Un connecteur d'entrée accepte *une seule* connexion. Si vous tentez d'ajouter une deuxième connexion à la même entrée, MapForce vous demandera de remplacer la connexion par une nouvelle ou de dupliquer l'item d'entrée[40]. Un connecteur de sortie peut avoir plusieurs connexions, chacune vers une entrée différente.

Pour copier une connexion vers un item différent, appuyez et tenez appuyée la section grasse à la fin de la connexion (*voir la capture d'écran dans Déplacer une connexion*) et glissez-la vers la destination sélectionnée tout en tenant la clé **Ctrl** appuyée.

Pour supprimer un connexion, cliquez sur la connexion et appuyez sur la touche **Supprimer**. De manière alternative, cliquez avec la touche de droite sur la connexion et sélectionnez **Supprimer** dans le menu contextuel.

Entrées obligatoires

Pour vous aider dans le processus de mappage, MapForce marque les entrées obligatoires en orange dans les composants cibles. L'exemple ci-dessous vous montre que dès que vous vous connectez à l'élément `book` du composant `Books` de l'élément `publication` du composant `BookOutput`, les connecteurs des nœuds obligatoires du composant `BooksOutput` seront mis en surbrillance. Si vous ne connectez pas d'entrées obligatoires, les nœuds respectifs ne seront pas mappés vers le cible et le mappage sera invalide.



Connexions parent manquantes

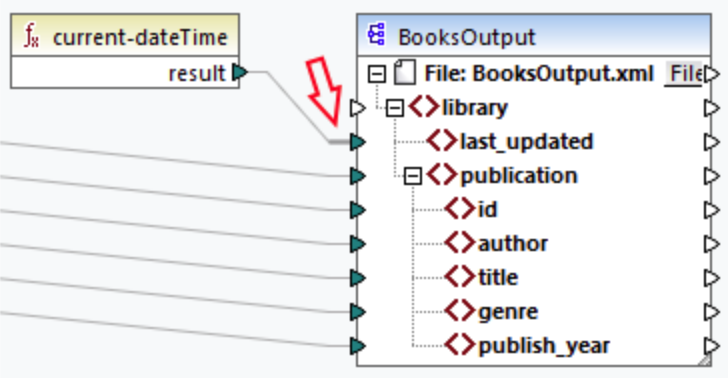
Lorsque vous créez des connexions entre les nœuds source et cible manuellement, MapForce analyse automatiquement les résultats possibles de mappage. Si vous vous connectez aux deux nœuds enfant sans les connecter à leurs nœuds parent, vous verrez un message de notification qui suggère la connexion du parent du nœud source au parent du nœud cible. Ce message de notification vous aide à éviter des situations dans lesquelles un seul nœud enfant apparaît dans le volet **Sortie**.

Si vous souhaitez déactiver de telles notifications, suivez les étapes suivantes :

1. Allez au menu **Outils** et cliquez sur **Options**.
2. Ouvre le groupe **Messages**.
3. Effacez la case à cocher **quand vous créez une connexion, suggérez la connexion d'items ancêtres**.

Déplacer une connexion

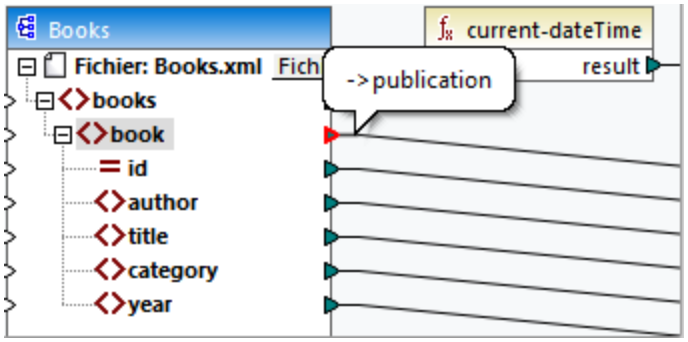
Pour déplacer une connexion vers un nœud différent, appuyez et tenez appuyée la section épaisse à la fin de la connexion (*voir la capture d'écran ci-dessous*) et glissez-la vers la destination sélectionnée.



Voir les info-bulles de connexion

Les info-bulles de connexion vous permettent de voir les noms des (i) nœuds auxquels les données sont mappées ou (ii) les nœuds depuis lesquels les données sont mappées. Pour voir les conseils, appuyez sur le bouton de la barre d'outils (**Afficher conseils**). Pour voir les noms des nœuds auxquels les données sont mappées, pointez le curseur sur la section épaisse d'une connexion près du connecteur de sortie (*voir la capture d'écran ci-dessous*). Pour voir le nom du nœud duquel les données sont mappées, pointez le curseur sur la section épaisse d'une connexion près d'un connecteur d'entrée. Si de multiples connexions ont été définies depuis la même sortie, un maximum de dix noms d'item seront affichés dans l'info-bulle.

Dans l'exemple ci-dessous, le nœud cible duquel les données de l'élément `book` sont mappées est appelé `publication`.



Changer les paramètres de connexion

Pour changer les paramètres de connexion, suivez une des étapes suivantes :

- Sélectionner une connexion. Puis, allez au menu **Connexion** et cliquez sur **Propriétés**.
- Double-cliquer sur la connexion.

- Cliquez avec la touche de droite sur la connexion, puis cliquez sur **Propriétés**.

Pour plus d'informations, voir Paramètres de connexion 56.

Mettre en surbrillance les connexions de manière sélective

MapForce vous permet de marquer les connexions de manière sélective dans un mappage. Cette fonction peut être utile quand votre mappage a de nombreux composants avec de multiples connexions. Marquer des connexions rendra la vérification plus facile, c'est-à-dire vérifier si les nœuds du composant sélectionné sont mappés correctement. Veuillez noter que le terme *connecteur* utilisé pour les touches de la barre d'outils se réfère à une connexion, c'est-à-dire une ligne connectant des nœuds de composant. Voir les options disponibles ci-dessous.

	**Afficher des connecteurs de composant sélectionnés** (seulement des connexions directes)
	**Afficher des connecteurs de la source vers le cible** (connexions directes et indirectes)

Uniquement des connexions directes

Quand le bouton **Direct-connections** n'est *pas* appuyé, vous pouvez voir toutes les connexions en noir. Quand le bouton **Direct-connections** est appuyé, seules les connexions liées au composant sélectionné actuellement sont en noir. Les autres connexions sont en gris clair.

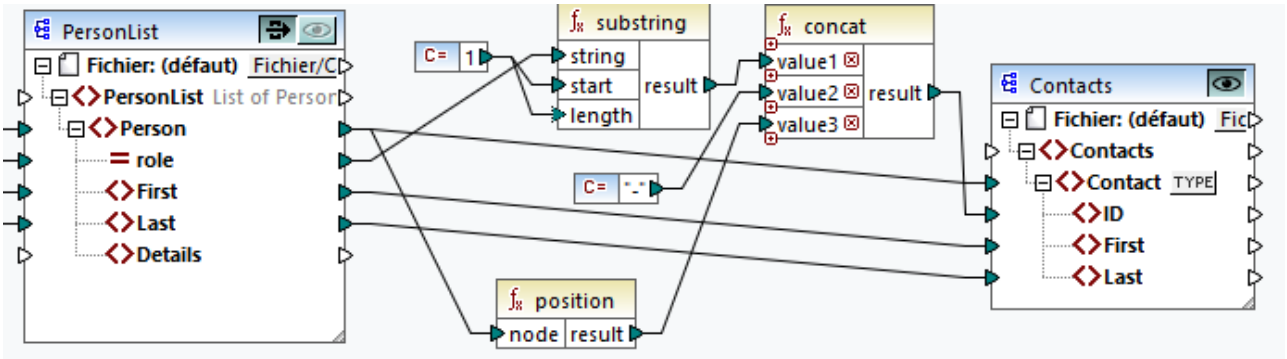
Connexions directes et indirectes

Le bouton **Connexions source-à-cible** devient disponible uniquement quand le bouton **Direct-connections** est appuyé. Quand le bouton **Connexions source-à-cible** est appuyé, vous pouvez tracer des connexions du composant sélectionné actuellement, y compris ses connexions directes et les connexions de ses composants connectés jusqu'aux fichiers source et cible.

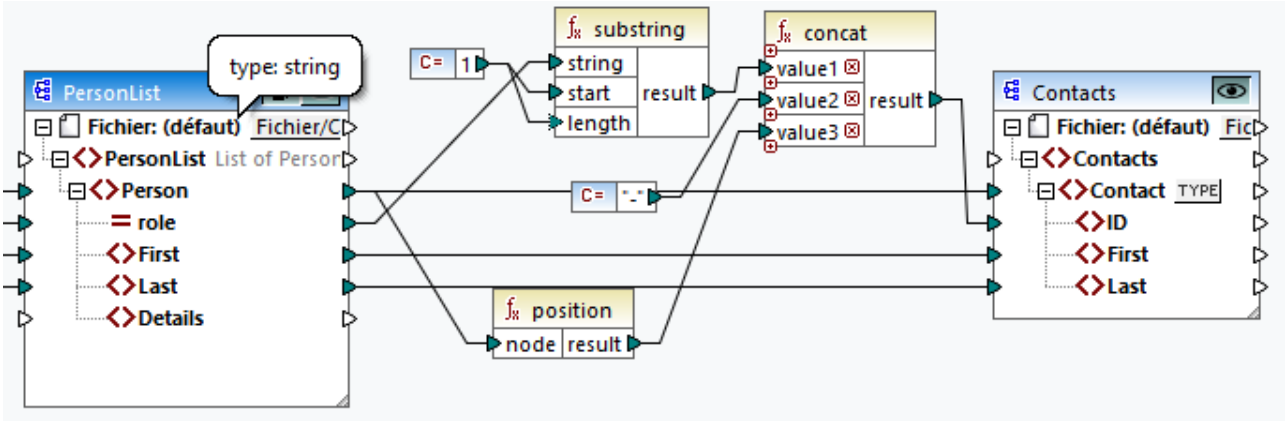
Pour comprendre comment ces deux options fonctionnent, voir l'exemple ci-dessous.

Exemple

La capture d'écran ci-dessous illustre la partie du mappage **`ChainedPersonList.mfd`**, qui est disponible dans le dossier **`MapForceExamples`**. Dans le mappage ci-dessous, nous avons appuyé sur le bouton **Direct-connections**, cliqué sur l'en-tête du composant **`concat`**, mais nous n'avons pas encore appuyé sur le bouton **Connexions source-à-cible**. Pour cette raison, nous voyons que seules les connexions directes de la fonction **`concat`** à la **`"-"`** constante, au `substring`, à la `position`, et aux composants `Contacts` sont noirs. Les autres connexions dans le mappage sont en gris clair.



La prochaine étape est d'appuyer sur le bouton **Connexions source-à-cible**. La capture d'écran ci-dessous reflète les changements :



Avec le bouton **Connexions source-à cible** appuyé, d'autres connexions sont devenus noires : (i) les connexions de la fonction du **`substring`** à la constante **`1`** et du composant `PersonList`, et (ii) la connexion de la fonction **`position`** au composant `PersonList`. Toutefois, les connexions entre le composant `PersonList` et son composant précédent restent gris clair. Donc, quand vous appuyez sur le bouton **Connexions source-à-cible** et cliquez sur le composant, vous pouvez retracer les connexions directes du composant. Si le composant sélectionné est connecté à quelques composants de transformation[33] (par ex., les fonctions, constantes, filtres, composants de tri, composants SQL-NoSQL-WHERE/ORDER, conditions if-else, value-maps), vous pourrez également voir leurs connexions jusqu'aux composants de structure[32] (tels que `PersonList` ci-dessus), variables, composants join, ou fonctions de service Web auxquels ces composants de transformation sont connectés.

Dans cette section

Cette section donne un aperçu des connexions et est organisée comme suit :

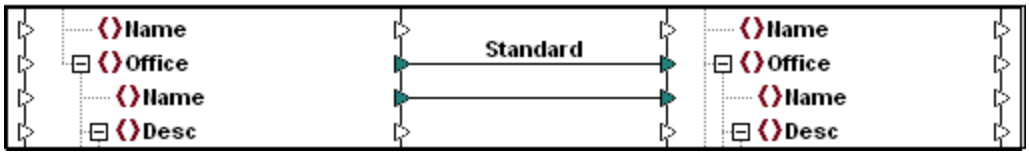
2.2.1 Types de connexion

Les types de connexion suivants sont disponibles dans MapForce :

- Connexions orientées vers la cible[50] (Standard) ;
- Connexions orientées vers la source[50] (Contenu mixte) ;
- Connexions d'enfants correspondants[52]
- Connexions copy-all[55] (Copier les items d'enfant).

Connexions orientées vers la cible par rapport aux connexions orientées vers la source

Les connexions orientées vers la cible et les connexions orientées vers la source sont mutuellement exclusives. Le choix entre ces deux options dépend de l'ordre dans lequel les nœuds ont besoin d'être mappés. Dans les connexions orientées vers la cible, l'ordre des nœuds dans la sortie est déterminé par le schéma *cible*. Ce type de connexion est adapté pour la plupart des scénarios de mappage et est le type de connexion par défaut utilisé dans MapForce. Une connexion orientée vers la cible est affichée en tant que ligne pleine (*voir la capture d'écran ci-dessous*).



Les connexions orientées vers la cible ne sont éventuellement pas adaptées quand vous voulez mapper des nœuds XML avec du contenu mixte (nœuds enfant et texte). Dans ce cas, une connexion orientée vers la source 50 est recommandée : L'ordre des nœuds dans la sortie est déterminée par le schéma *source*.

Enfants correspondants et connexions copy-all

Les connexions d'enfants correspondants et connexions copy-all appartiennent à une sous-section de connexions orientées vers la cible et la cible. Les enfants correspondants et connexions copy-all mappent des données entre les nœuds avec des nœuds enfant qui sont semblables dans les composants source et cible. Les connexions copy-all sont semblables aux connexions d'enfants correspondants, mais ne sont dotées que d'une connexion épaisse à la place de multiples connexions, qui empêche la zone de mappage d'être visuellement encombrée.

La section fournit des informations sur chaque type de connexion et sur les scénarios quand ces types de connexion sont utiles.

2.2.1.1 Connexions orientées vers la source

Une connexion orientée vers la source vous permet de mapper le contenu mixte (nœuds texte et enfant) dans le même ordre que dans le fichier XML *source*. Une connexion de contenu mixte est affichée en tant que ligne pointillée au niveau du nœud parent (*voir le Mappage l'élément <para>*). Cette rubrique explique comment mapper du contenu mixte. Elle affiche également l'effet d'utiliser les connexions standard (orientées vers la cible) avec du contenu mixte.

Note : les connexions orientées vers la source peuvent également être utilisées dans les champs de base de données avec du contenu mixte (*éditions Professional et Enterprise*).

Note : en vue d'accepter du contenu mixte, les composants cible doivent avoir des nœuds de contenu mixte.

Mappage de contenu mixte

Cette rubrique explique comment mapper le contenu mixte en utilisant une connexion axée sur la source Vous aurez besoin des fichiers suivants : **`Tut-OrgChart.xml`**, **`Tut-Orgchart.mfd`**, **`Tut-Person.xsd`**, et **`Tut-OrgChart.xsd`**, qui sont disponibles dans le dossier du Tutoriel 16.

Instance XML source

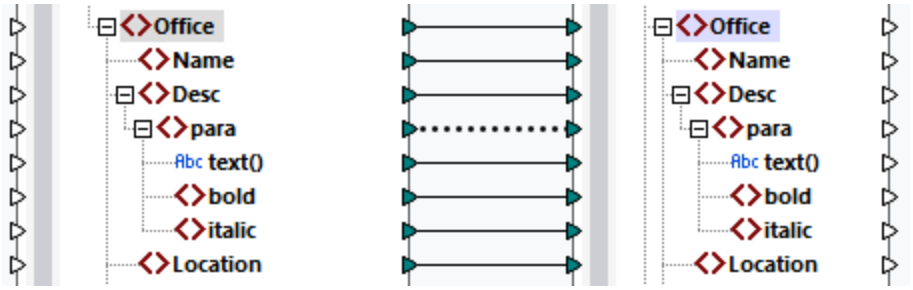
Un snippet du **Tut-OrgChart.xml** est affiché ci-dessous. Dans cet exemple, nous nous concentrerons sur l'élément du contenu mixte <para> avec ses nœuds enfant <bold> et <italic>. L'élément <para> contient aussi une instruction de traitement (<?sort alpha-ascending?>) ainsi qu'un commentaire (<!--Company details... -->), qui peuvent aussi être mappés, comme indiqué ci-dessous. Veuillez noter la séquence du texte et des nœuds gras/italique dans le fichier d'instance XML.

```
 8        <Desc>
 9          <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.
10           <?sort alpha-ascending?>
11           <!--Company details: location and general company information.-->
12         </para>
13         <para>White papers and further information will be made available in the near future.
14         </para>
15       </Desc>
```

Mapper l'élément <para>

L'image ci-dessous illustre une partie de **Tut-Orgchart.mfd**. Dans l'exemple ci-dessous, la ligne pointillée montre que l'élément <para> contient du contenu mixte. Pour créer des connexions de contenu mixte, suivez les étapes suivantes :

1. Sélectionnez la commande de menu **Connexion | Auto-connexion des enfants correspondants**, qui se connectera aux nœuds enfant correspondant 52 automatiquement. De manière alternative, vous pouvez mapper manuellement le nœud <para> avec ses nœuds enfant.
2. Connectez l'élément <para> dans le composant source avec l'élément <para> dans le composant cible. Un message vous demandera si vous souhaitez définir la connexion en tant que « orientée vers la source ».
3. Cliquez **Oui** pour créer une connexion à contenu mixte.
4. Cliquez sur le volet **Sortie** pour voir le résultat du mappage. Cliquez sur le bouton  (**Wrap**) dans la barre d'outils du volet **Sortie** pour afficher la liste de code entière (par ex., n'allant pas au-delà de la barre de défilement) dans le volet **Sortie**. Le contenu mixte du nœud <para> a été mappé dans le même ordre qu'il apparaît dans le fichier source XML.



Traiter les instructions et les commentaires

Si votre mappage a des instructions de traitement et/ou commentaires et que vous voulez les mapper, suivez les étapes ci-dessous :

1. Cliquez de la touche de droite sur la connexion du contenu mixte (ligne pointillée) et sélectionnez **Propriétés**.
2. Sous **Source-Drive (contenu mixte)**, sélectionnez les cases à cocher **Mapper les instructions de traitement** et/ou les **Commentaires de mappage**.

Connexions orientées vers la cible avec du contenu mixte
Choisir des connexions orientées vers la cible pour du contenu mixte peut avoir des conséquences indésirables. Pour voir comment des connexions orientées vers la cible affectent l'ordre des nœuds à contenu mixte, suivez la instructions suivantes :

1. Ouvrez **`Tut-ExpReport.mfd`** dans le **`dossier du Tutoriel`**.
2. Cliquez sur le bouton de barre d'outils (Auto-connexion des enfants correspondants 52). Supprimez la case à cocher **Créer des connexions copy-all** dans les pour les connexions d'enfants correspondants 52. Ceci empêchera MapForce de créer les connexions copy-all 55 automatiquement.
3. Créer une connexion entre le nœud `para` dans la source et le nœud `para` dans la cible. Un message vous demandera si vous souhaitez définir les connexions en tant que « orientées vers la source ». Cliquez sur **Non**. Ceci crée une connexion orientée vers la cible.
4. Cliquez sur le volet **Sortie** pour voir le résultat du mappage (*la capture d'écran ci-dessous*).

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

La capture d'écran ci-dessus monte que le contenu de l'élément `text()` dans la source a été mappé vers la cible. Toutefois, l'ordre des nœuds enfant (`gras` et `italique`) dans la sortie correspond à l'ordre de ces nœuds dans le schéma XML cible. Ceci signifie que les éléments `gras` et `italiques` ne sont pas intégrés dans le texte, mais sont mappés séparément.

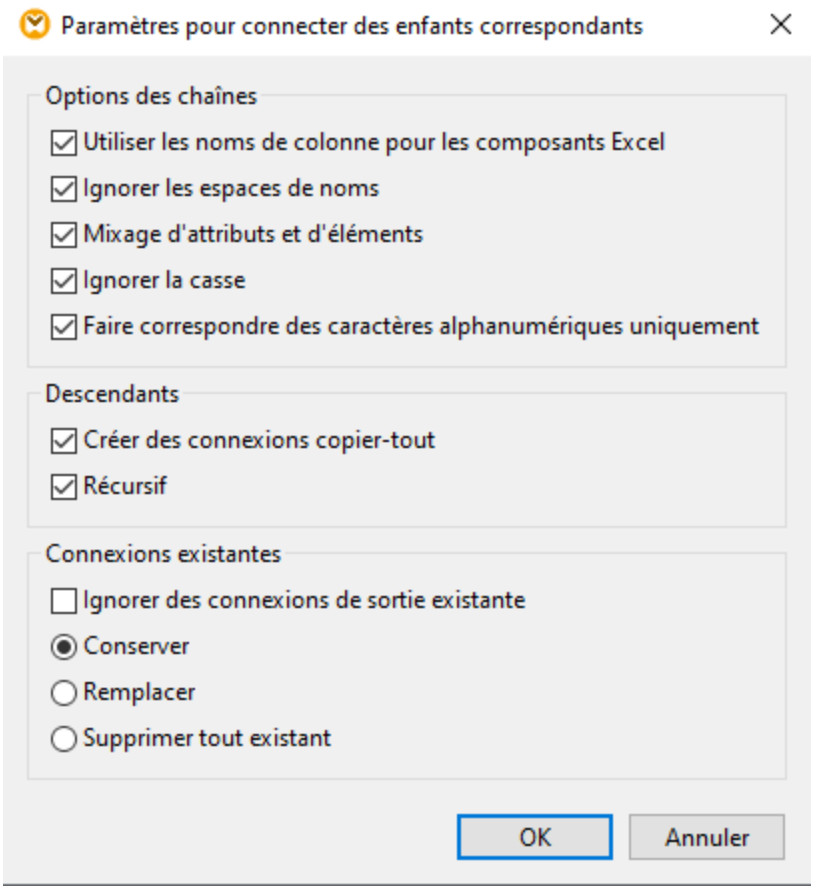
2.2.1.2 Connecter des enfants correspondants

Les connexions d'enfants correspondants connectent automatiquement tous les nœuds enfant qui sont dotés des mêmes noms dans les fichiers source et cible. Pour activer cette option, procédez comme suit :

- Cliquez sur le bouton de barre d'outils (**Auto-connexion des enfants correspondants**).
- Allez au menu **Connexion** et cliquez sur **Auto-connexion des enfants correspondants**.

Paramètres pour les connexions d'enfants correspondants

Pour configurer les paramètres pour les connexions d'enfants correspondants, cliquez avec la touche de droite sur toute connexion et sélectionnez l'option **Connecter les enfants correspondants** du menu contextuel ou allez au menu **Connexion** et cliquez sur **Paramètres pour Connecter les enfants correspondants**. Ceci ouvre le dialogue **Paramètres pour Connecter les enfants correspondants** *capture d'écran ci-dessous*).



La liste ci-dessous décrit les options disponibles dans la boîte de dialogue **Paramètres pour Connecter des enfants correspondants**. Les paramètres dans la boîte de dialogue ne s'appliquent que si le  bouton de barre d'outils (**Toggle auto-connexion des enfants**) est appuyé.

Options correspondantes

La section *Options correspondantes* vous permet de détendre les critères correspondants et de définir comment comparer des noms de nœuds. Les options suivantes sont disponibles :

- *Utilisez les noms de colonne pour des composants Excel :* Cette option s'applique uniquement aux composants Excel (*Enterprise Edition*). Cette option signifie que les noms de colonne définis par l'utilisateur (par ex., `Company`) seront utilisés pour comparaison à la place des noms de référence de colonnes (par ex., A, B, C). Les noms de colonne définis par l'utilisateur sont définis dans le dialogue **Sélectionner des plages de cellules** et apparaissent comme annotations dans le composant Excel.
- *Ignorer les espaces de noms :* Les enfants correspondants seront connectés indépendamment des espaces de noms des nœuds d'enfant.
- *Mixer les attributs et les éléments :* Cette option permet la création de connexions entre les attributs et les éléments qui portent les mêmes noms. Par exemple, une connexion est créée si deux nœuds `ID` existent, même si un est un élément et l'autre un attribut.
- *Ignorer la casse :* Les enfants correspondants seront connectés indépendamment de la casse des noms de nœuds d'enfant.

- *Faire correspondre un caractère alpha-numérique uniquement :* Lorsque cette option est activée, seuls les chiffres et lettres seront comparés. D'autres caractères tels que les espaces, virgules, points etc. ne seront pas abandonnés avant la comparaison.

Descendants
La section *Descendants* définit comment procéder aux nœuds enfant. Les options suivantes sont disponibles :

- *Créer des connexions copier-tout :* Ce paramètre est actif par défaut. Il crée (si possible) une connexion copy-all[55], qui mappe les données entre des nœuds enfant étant similaires ou identiques. Une connexion copy-all est représentée par une ligne épaisse, qui prévient le désordre et rend la compréhension du mappage plus facile.
- *Récursif :* Cette option crée de nouvelles connexions entre les nœuds correspondants s'ils ont les mêmes noms. Et cela n'a aucune importance à quel niveau les nœuds sont imbriqués dans l'arborescence.

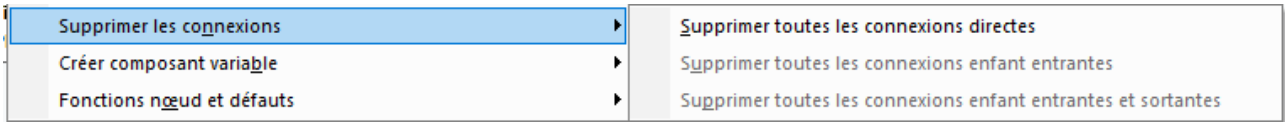
Connexions existantes
La section *Connexions existantes* précise quoi faire des connexions existantes. Les options suivantes sont disponibles :

- *Ignorer les connexions de sortie existantes :* Cette option crée des connexions supplémentaires pour tous les nœuds correspondants, même s'ils ont déjà des connexions sortantes.
- *Retenir :* Cette option retient les connexions existantes.
- *Écraser :* Cette option écrase les connexions existantes.
- *Supprimer tout existant :* Cette option supprime toutes les connexions existantes avant d'en créer de nouvelles.

Supprimer des connexions en tant que groupe

Si vous souhaitez supprimer des connexions en tant que groupe, suivez les instructions ci-dessous :

1. Clic droit sur un nom de nœud dans le composant.
2. Sélectionnez **Supprimer des connexions | Supprimer toutes <...> connexions** depuis le menu contextuel (*voir la capture d'écran ci-dessous*).



- *Supprimer toutes les connexions directes :* Cette option supprime toutes les connexions qui sont directement mappées vers ou depuis des nœuds sélectionnés.
- *Supprimer Toutes les connexions enfant entrantes :* Cette option est active uniquement si vous avez cliqué avec la touche droite sur un nœud parent dans un composant cible. Cette option supprime toutes les connexions enfant entrantes du nœud parent sélectionné.
- *Supprimer Toutes les connexions enfant sortantes :* Cette option est active uniquement si vous avez cliqué avec la touche droite sur un nœud parent dans un composant source. Cette option supprime toutes les connexions enfant du nœud parent sélectionné.

2.2.1.3 Connexions copier tout

Les connexions copier-tout mappent les données entre des nœuds avec des items enfant qui sont similaires ou identiques. Les connexions « Copier tout » ne sont possibles que pour des formats identiques (par ex., JSON à JSON ou XML à XML). Ce principe s'applique également à tous les composants texte : fichiers plats, fichiers FlexText et EDI. Puisqu'il s'agit de fichiers texte pour ces formats, vous pouvez les combiner et créer une connexion copy-all entre les fichiers EDI et FlexText, par exemple.

Le principal avantage des connexions copier-tout est qu'elles simplifient visuellement l'espace de travail du mappage : Une connexion représentée par une ligne épaisse est créée à la place de multiples connexions (*voir l'exemple dans Créer toutes les connexions copier-tout manuellement*). Les sous-sections ci-dessous expliquent comment créer des connexions copy-all automatiquement et manuellement.

Créer des connexions copy-all automatiquement

Pour créer une connexion copy-all automatiquement, suivez les étapes suivantes :

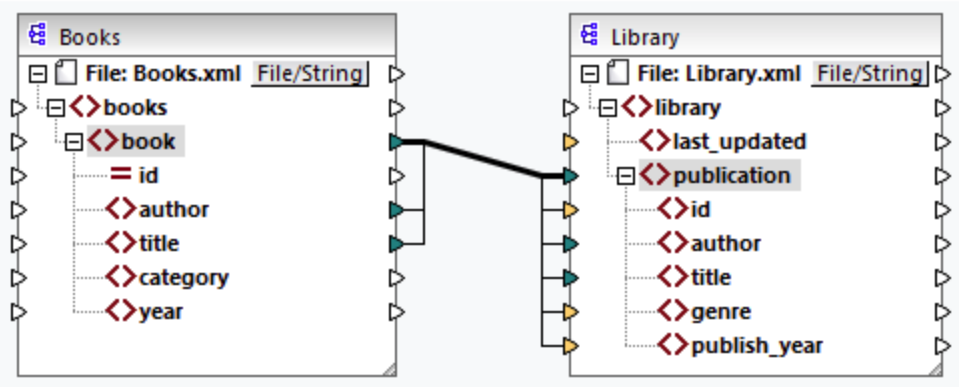
1. Allez au menu **Connexion**.
2. Cliquez sur **Paramètres pour connecter des enfants correspondants**.
3. Vérifiez la case **Créer des connexions copier-tout** et cliquez sur **OK**.
4. Appuyez sur la touche de la barre d'outils **Toggle auto-connexion d'enfants**. De manière alternative, allez au menu **Connexion** et cliquez sur **Auto-connexion des enfants correspondants**.

Si des types et/ou des noms de nœuds enfant dans la source et la cible ne sont pas les mêmes, une connexion copy-all ne sera pas créée automatiquement, et vous devrez en créer une automatiquement.

Créer des connexions copier-tout manuellement

Pour créer une connexion copy-all manuellement, suivez les étapes suivantes :

1. Ajouter un fichier source : Cliquez sur **XML Schema/File** dans le menu **Insert** et recherchez **`Books.xml`** situés dans le dossier BasicTutorials 16.
2. Ajouter un fichier cible : Cliquez sur **XML Schema/File** dans le menu **Insérer** et recherchez **`Library.xsd`** situé dans le même dossier que **`Books.xml`**. Cliquez sur **Ignorer** lorsque vous êtes invité par MapForce à ajouter un exemple de fichier XML.
3. Mappez le nœud `<book>` du composant **`books`** au nœud `<publication>` du composant **`library`**. Comme la structure des éléments de `<book>` et `<publication>` ne coïncide pas entièrement, la connexion copy-all n'est pas créée. À la place, la fonction **Auto-connexion des enfants correspondants** connecte automatiquement tous les nœuds enfant avec le même nom, ce qui est expliqué dans le Tutoriel 1 83.
4. Pour modifier la connexion automatique à une connexion copy-all, cliquez avec la touche droite sur la connexion entre `<book>` et `<publication>` et sélectionnez **Copy-All (copiez les éléments enfant)** depuis le menu contextuel.
5. Une fenêtre pop-up proposera de remplacer les connexions existantes par une connexion copy-all. Cliquez sur **OK**. Maintenant la source et la cible ont une connexion copy-all (*voir la capture d'écran*).



Dans le mappage ci-dessus, seuls deux nœuds enfant sont identiques dans les deux structures : `<author>` et `<title>`. Pour cette raison, une connexion copy-all existe entre ces nœuds. Les nœuds enfant qui ne sont pas les mêmes ne peuvent pas être connectés. La capture d'écran montre que l'`id` n'est pas incluse dans la connexion copy-all car son type n'est pas le même dans la source et dans la cible : l'`id` est un attribut dans la source et un élément dans la cible. Si vous tentez de créer une connexion entre les éléments qui ne sont pas les mêmes, par ex., `<category>` et `<genre>`, MapForce vous invitera à la remplacer ou à dupliquer l'entrée (voir la capture d'écran ci-dessous).

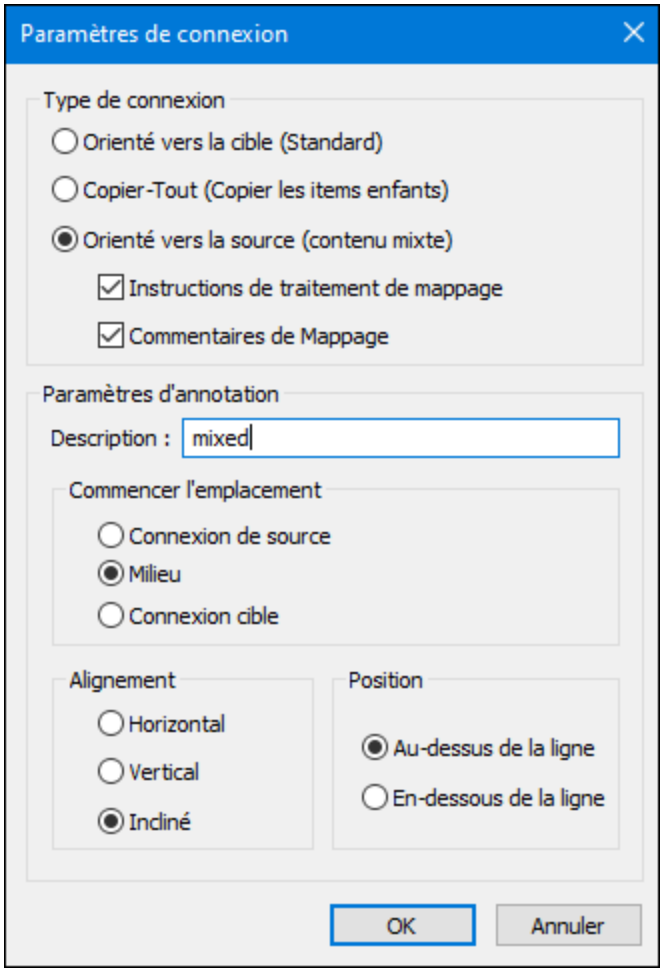
La duplication de l'entrée[40] n'a de sens que si vous souhaitez que la cible accepte des données depuis plus d'une entrée, ce qui n'est pas nécessaire ici. Si vous choisissez de remplacer la connexion "Copier-tout", un message vous invite à nouveau soit à résoudre, soit à supprimer la connexion "Copier-tout". Cliquez sur **Résoudre la connexion copier-tout** si vous souhaitez remplacer la connexion copier-tout par des connexions orientées vers la cible[50] individuelles. Si vous préférez supprimer entièrement la connexion copy-all, cliquez sur **Supprimer connexions enfant**.

Important

Lors de la création de connexions copier-tout entre un schéma et un paramètre d'une fonction définie par l'utilisateur[203], les deux composants doivent être basés sur le même schéma. Néanmoins, il n'est pas nécessaire qu'ils possèdent les mêmes éléments racine.

2.2.2 Paramètres de connexion

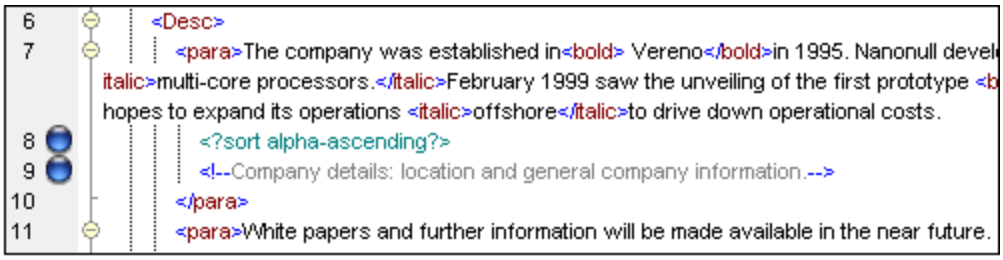
La boîte de dialogue **Paramètres de connexion** définit les paramètres d'une connexion. Pour ouvrir cette boîte de dialogue, double-cliquez sur la connexion. De manière alternative, cliquez avec la touche de droite sur une connexion et sélectionnez **Propriétés** depuis le menu contextuel. Les paramètres sont divisés en deux parties : les types de connexion et les paramètres d'annotation. Pour plus d'information, veuillez voir les sous-sections ci-dessous.



Types de connexion

Vous pouvez choisir un des types de connexion décris ci-dessous:

- Les connexions orientées vers la cible (Standard) 50 sont adaptées pour la plupart des scénarios de mappage.
- Les connexions Copy-all (Copier tous les items enfant) 55 : Si des composants source et cible ont des nœuds identiques ou similaires avec des nœuds enfant correspondants, une connexion copy-all sera automatiquement créée entre les nœuds correspondants.
- Les connexions orientées vers la source (contenu mixte) 50 mappent le contenu mixte (nœuds texte et enfant) dans le même ordre que dans le fichier XML source. Si vous sélectionnez **Mapper les instructions de traitement** et/ou **Mapper les commentaires**, vous pourrez inclure ces groupes de données dans le fichier sortie (*voir la capture d'écran ci-dessous).*).



Paramètres d'annotation

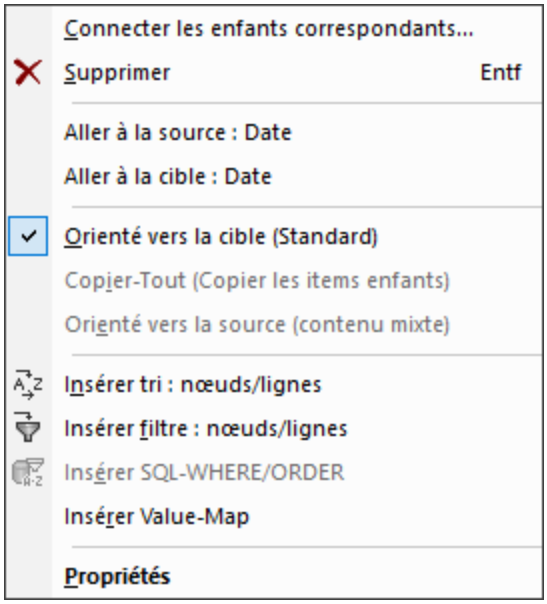
La section des **paramètres d'annotation** section vous permet d'étiqueter la connexion. Cette option est disponible pour tous les types de connexion. Pour annoter une connexion, suivez les instructions ci-dessous :

1. Cliquez avec la touche de droite sur la connexion et sélectionnez **Propriétés** depuis le menu contextuel. En alternative, double-cliquez sur la connexion.
2. Saisissez le nom de la connexion sélectionnée actuellement dans le champ **Description**. Cela active toutes les options dans la section **Paramètres d'annotation**.
2. Utilisez les groupes restants pour définir les paramètres de **démarrage emplacement**, **alignement** et **position** de l'étiquette.
3. Appuyez sur le bouton de la barre d'outils (**Afficher les annotations**). Si le bouton n'est pas encore visible dans la barre d'outils, activez le bouton dans la barre d'outils **Afficher les options**.

Note : si la touche de la barre d'outils **Afficher annotations** est inactive, vous pouvez toujours consulter l'annotation si vous placez le curseur de la souris sur la connexion. L'annotation apparaîtra comme info-bulle si la touche de la barre d'outils (**Afficher les conseils**) est active dans la barre d'outils **Afficher les options**.

2.2.3 Menu contextuel de la connexion

Cette rubrique décrit les commandes disponibles dans le menu contextuel de la connexion. Lorsque vous cliquez avec la touche de droite sur une connexion, les commandes contextuelles suivantes sont disponibles :



Pour plus d'information, veuillez voir les sous-sections ci-dessous.

Paramètres généraux

- *Connecter les enfants correspondants :* Ouvre la boîte de dialogue Connecter les enfants correspondants[52]. Cette commande est activée lorsque la connexion est susceptible d'avoir des enfants correspondants.
- *Supprimer :* Supprime la connexion sélectionnée.
- *Aller à la source : <item name>* Met en surbrillance le connecteur de sortie[31] de la connexion sélectionnée.
- *Aller à la cible : <item name>* Met en surbrillance le connecteur d'entrée[31] de la connexion sélectionnée.

Types de connexion

Voir les détails sur les types de connexion dans Types de connexion[49] et Paramètres de connexion[56].

Commandes d'insertion

- *Insérer Tri : Nœuds/Lignes :* Ajoute un composant de tri[170] entre un nœud source et un nœud cible.
- *Insérer Filtre : Nœuds/Lignes :* Ajoute un composant de filtre[176] entre un nœud source et un nœud cible.
- *Insérer SQL/NoSQL-WHERE/ORDER:* Ajoute un composant SQL/NoSQL-WHERE/ORDER entre un nœud source et un nœud cible (*éditions Professional et Enterprise*).
- *Insérer Value-Map :* Ajoute une value-map[182] entre le nœud source et le nœud cible.

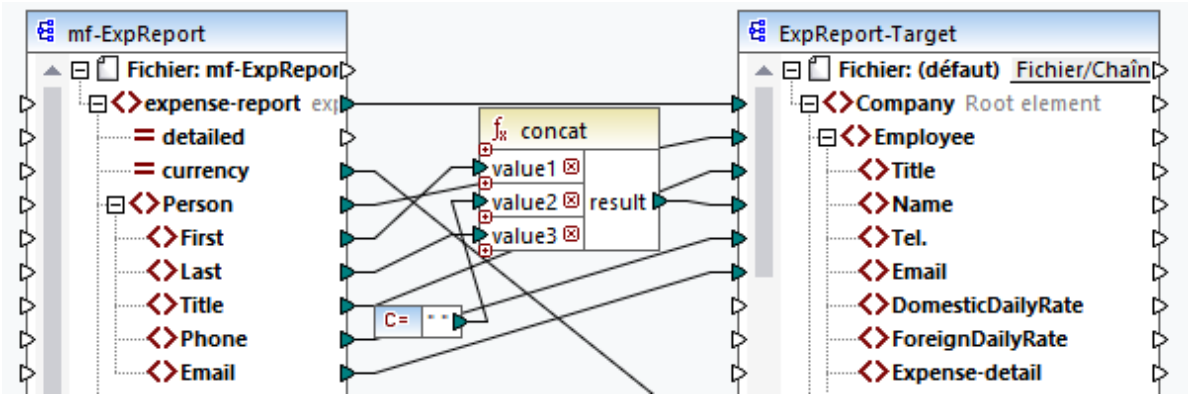
Propriétés

Ouvre la boîte de dialogue des Paramètres de connexion[56].

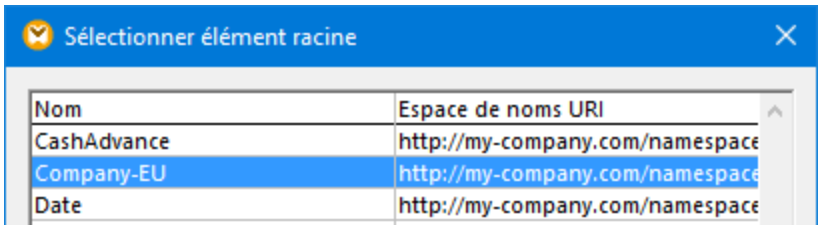
2.2.4 Connexions incorrectes

Il existe des situations dans lesquelles vous allez éventuellement vouloir changer un schéma d'une source ou d'une cible. Les changements à un schéma peuvent affectés la validité de leur mappage et le résultat dans plusieurs connexions incorrectes. La rubrique explique comment corriger de telles connexions après avoir changé le fichier de schéma. Suivez les instructions dans l'exemple ci-dessus pour comprendre comment gérer les connexions incorrectes.

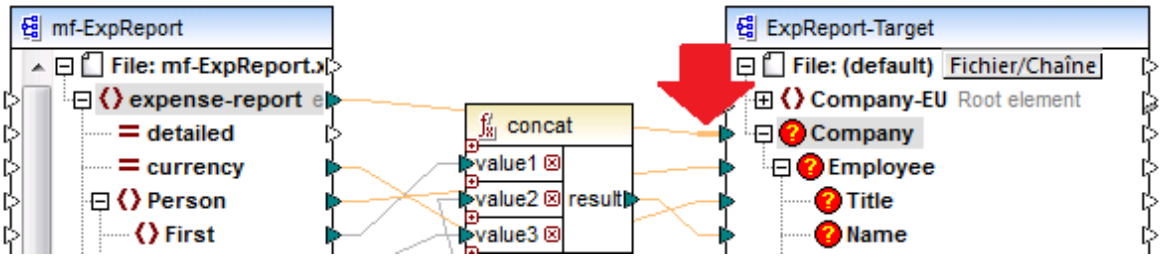
1. Ouvrez **`Tut-ExpReport.mfd`** disponible sous le dossier du Tutoriel[16]. La partie de ce mappage est affiché ci-dessous.



2. Ouvrez **`ExpReport-Target.xsd`** dans l'éditeur (par ex., Altova XMLSpy) et modifiez l'élément racine `Company` dans le schéma cible en `Company-EU`. Il n'est pas nécessaire de fermer MapForce.
3. Une fois avoir édité l'élément racine du schéma cible**,** l'invite des **fichiers Changés** apparaît dans MapForce. Cliquez sur la touche **Recharger**. Puisque l'élément racine a été supprimé, le composant affiche plusieurs nœuds incorrects.
4. Cliquez sur **Sélectionner un nouvel élément racine** dans la partie supérieure du composant (*voir la capture d'écran ci-dessous*). Vous pouvez aussi changer l'élément racine en cliquant avec la touche de droite sur l'en-tête du composant et en sélectionnant **Changer l'élément racine** depuis le menu contextuel.



5. Sélectionnez `Company-EU` en tant que nouvel élément racine et cliquer sur **OK**. L'élément racine `Company-EU` est maintenant visible dans la partie supérieure du composant.
6. Maintenant, vous devez déplacer la connexion du nœud incorrect `Company` au nouvel élément racine. Appuyez et tenez appuyée la section épaisse (*voir la flèche rouge ci-dessous*) de la connexion de `Company`. Puis glissez la connexion vers l'élément racine `Company-EU`.



Une boîte de dialogue de notification vous demandera si vous souhaitez déplacer tous les nœuds enfant connectés correspondants. Vous pouvez choisir entre uniquement vouloir déplacer la connexion sélectionnée ou la connexion sélectionnée avec ses nœuds enfant qui correspondent aux nœuds enfant du nouvel élément racine. Dans notre exemple, nous avons choisi l'option **Inclure des connexions décroissantes**. Dès que vous cliquez sur le bouton, tous les nœuds incorrects disparaîtront du composant.

Note : si le nœud vers lequel vous mappez porte le même nom que le nœud source, mais a un espace de noms différent, le dialogue de notification contiendra une touche supplémentaire **Inclure les espaces de noms décroissants et mapper l'espace de noms**. Cliquez sur cette touche pour déplacer les connexions enfant du même espace de noms en tant que nœud parent de source vers les mêmes nœuds enfant sous le nœud d'espace de noms différent.

Solution alternative

Une solution alternative au problème discuté ci-dessus pourrait supprimer les nœuds incorrects dont vous n'avez éventuellement plus besoin dans votre mappage. Par exemple, si vous supprimez la connexion entre la fonction `concat` et le nœud `Nom`, le `Nom` disparaîtra du composant `ExpReport-Target`.

Les connexions incorrectes dans les bases de données (éditions Professional et Enterprise)

Si votre base de données a des connexions incorrectes, vous allez devoir changer les paramètres de composant [39] Cliquer sur le bouton **Changer** dans la boîte de dialogue **Paramètres de composant** vous permet de sélectionner une base de données différente ou de changer les tables dans votre composant de base de données. Toutes les connexions valides/correctes et les données de base de données pertinentes seront retenues si vous sélectionnez une base de données de la même structure.

2.2.5 Garder des connexions après avoir supprimé des composants

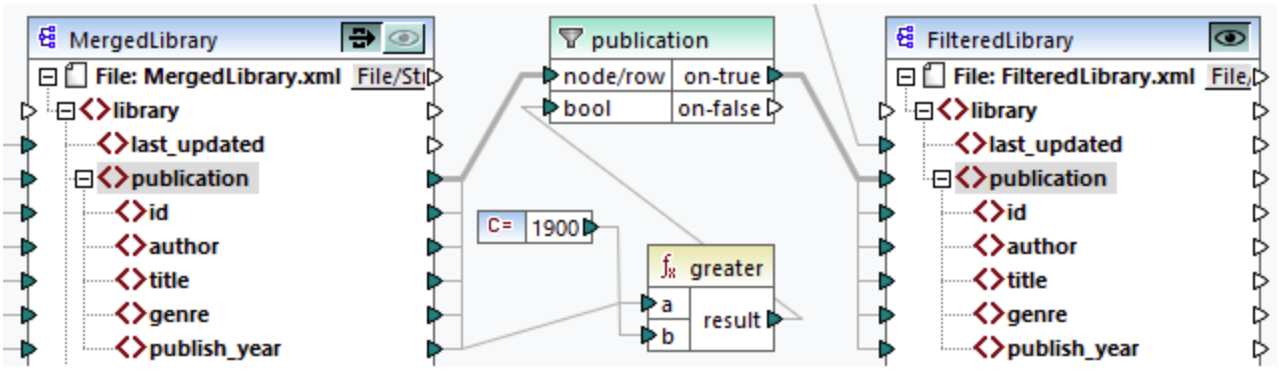
MapForce vous permet de garder les connexions même après avoir supprimé quelques composants de transformations [32] : par ex., les variables, composants de tri et de filtre, value-maps, entrées simples, composants SQL/NoSQL-WHERE/ORDER. Les connexions peuvent être simples ou multiples. Garder les connexions peut être particulièrement utile avec de multiples connexions enfant car vous ne devrez pas restaurer chaque connexion simple enfant manuellement après avoir supprimé le composant de transformation. Pour activer cette option, allez à **Outils | Options | Édition** et sélectionnez **Suppression de composant intelligent (garder les connexions utiles)**. Par défaut, cette option est désactivée, ce qui signifie que supprimer un composant de transformation supprimera également ses connexions directes.

Exemple

L'exemple de fichier appelé **`Tut3-ChainedMapping`** est utilisé pour illustrer une suppression de composant intelligent. L'exemple de fichier est disponible dans le dossier BasicTutorials 16.

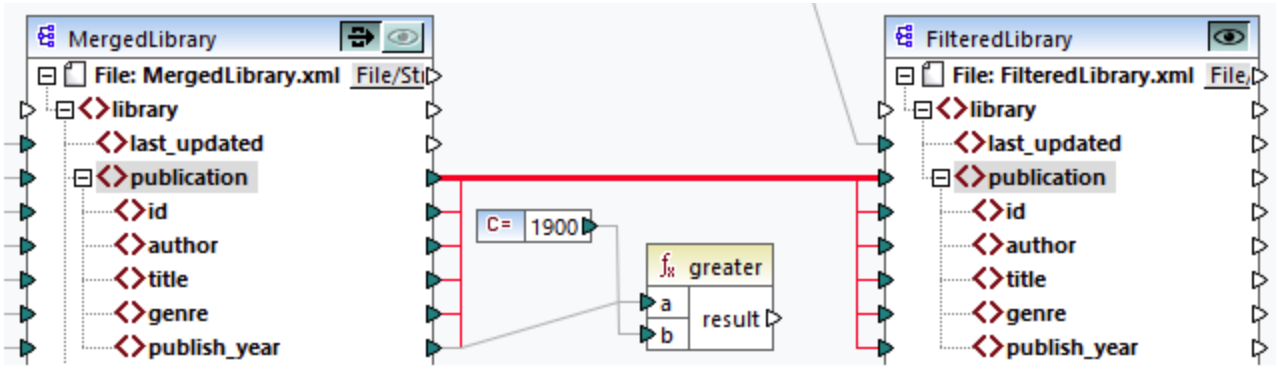
Avant la suppression

La capture d'écran ci-dessous montre que les connexions copy-all 55 existent entre le composant `MergedLibrary` et le filtre `publication`, et entre le filtre `publication` et le composant `FilteredLibrary`. Maintenant, nous voulons supprimer le filtre `publication` mais retenir les connexions copy-all. Pour ce faire, sélectionnez la case à cocher **Suppression de composant intelligent** dans la boîte de dialogue des **Options** (*voir la capture d'écran ci-dessus*).



Après la suppression

Après avoir supprimé la fonction `publication`, la connexion copy-all a été créée directement entre le nœud `publication` dans `MergedLibrary` et le nœud `publication` dans `FilteredLibrary` (*voir la capture d'écran ci-dessous*).



Note : si un composant de filtre est connecté aux deux sorties `on-true` et `on-false`, les connexions des deux sorties seront retenues.

2.3 Procédures générales et fonctions

Outre la création de mappages, vous pouvez également valider votre mappage et la sortie, générer le code, utiliser les fonctions du mode Texte et définir les paramètres de mappage. Cette section est organisée dans les rubriques suivantes :

2.3.1 Validation

Cette rubrique explique comment valider des mappages. Cette rubrique vous montre également comment prévisualiser, enregistrer et valider votre sortie.

Valider les mappages

MapForce valide les mappages automatiquement quand vous cliquez sur le volet **Sortie**. Vous pouvez aussi valider votre mappage manuellement, ce qui peut vous aider à identifier et corriger des erreurs potentielles avant d'exécuter le mappage. Pour valider un mappage manuellement, cliquez sur le volet **Mappage**, puis procédez à l'une des étapes suivantes :

- Cliquez sur **Valider mappage** dans le menu **Fichier**.
- Cliquez sur  (**Valider**) dans la barre d'outils.

Lorsque vous validez un mappage, MapForce vérifie, par exemple, des types de composants non pris en charge et des connexions incorrectes ou manquantes. Pour en savoir plus sur les statuts de validation, voir Fenêtre de messages 25. La fenêtre **Messages** vous permet également de prendre des actions liées aux messages 26. Pour afficher le résultat de chaque validation dans un onglet individuel, cliquez dans l'onglet numéroté disponible à gauche de la fenêtre **Messages**. Cela peut être utile, par exemple, si vous travaillez avec de multiples fichiers simultanément.

Validation des composants de transformation
La validation des composants de transformation 33 fonctionne comme suit :

- Si un **connecteur d'entrée** obligatoire est non-connecté, un message d'erreur est généré et la transformation est interrompue.
- Si un **connecteur de sortie** n'est pas connecté, un avertissement est généré et le processus de transformation se poursuit. Le composant, qui a causé l'avertissement, et ses données sont ignorés et ne sont pas mappés vers la cible.

Prévisualiser et valider la sortie

MapForce vous permet de prévisualiser la sortie sans avoir à exécuter et à compiler le code généré avec un processeur ou compilateur externe. En général, il est recommandé de consulter la sortie de transformation

dans le cadre de MapForce avant de traiter le code généré extérieurement. Lorsque vous consultez les résultats de mappage, MapForce exécute le mappage et affiche la sortie dans le volet de sortie[28].

Une fois que les données sont disponibles dans le volet **Sortie**, vous pouvez les valider et les enregistrer si nécessaire. Vous pouvez aussi utiliser la commande **Trouver** (**Ctrl + F**) pour situer rapidement un motif de texte particulier dans le fichier de sortie. Pour plus d'informations, voir Recherche Mode Texte[71]. Tout avertissement, erreur ou messages d'information lié à l'exécution de mappage est affiché dans la fenêtre Messages[25].

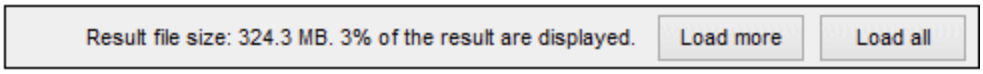
Si vous sélectionnez C++, C# ou Java (*éditions Professional et Enterprise*) en tant que langage de transformation[17], MapForce exécute le mappage utilisant son moteur de transformation built-in et affiche le résultat dans le volet **Sortie**.

Pour enregistrer la sortie de transformation, cliquez sur le volet **Sortie**, puis procédez à l'une des étapes suivantes :

- Cliquez sur **Enregistrer fichier de sortie** dans le menu **Sortie**.
- Cliquez sur  (**Enregistrer sortie générée**) dans la barre d'outils.

Charger les options

Lorsque vous consultez des fichiers de sortie volumineux, MapForce limite la quantité de données affichées dans le volet **Sortie**. Dans ce cas, le bouton **Charger plus** apparaît dans la partie inférieure du volet (*voir la capture d'écran ci-dessous*). Cliquer sur le bouton **Charger plus** ajoute le prochain morceau de données. Vous pouvez configurer les paramètres d'aperçu depuis l'onglet **Général** du dialogue **Options**. Pour plus d'informations, voir Options MapForce[469].

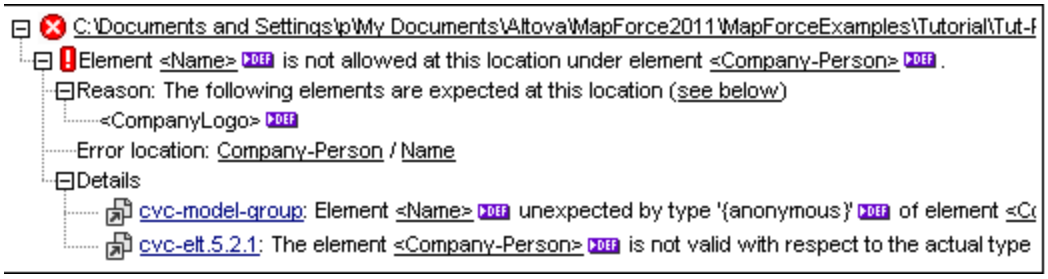


Valider la Sortie

Dès que la sortie devient disponible dans le volet **Sortie**, vous pouvez la valider par rapport au schéma y associé. Veuillez noter que la touche **Valider Sortie** et sa commande de menu correspondante (Sortie **| Valider fichier de sortie**) sont uniquement activées si le fichier de sortie prend en charge la validation par rapport à un schéma. Le résultat de la validation est affiché dans la fenêtre **Messages**, par exemple : Pour valider la sortie, suivez une des étapes suivantes :

- Ouvrez le volet **Sortie** et cliquez sur  (**Valider Sortie**) dans la barre d'outils.
- Ouvrez le volet **Sortie** et cliquez sur **Valider Fichier de sortie** dans le menu **Sortie**.

La capture d'écran ci-dessous illustre une validation qui n'a pas réussi. La fenêtre **Messages** contient des informations détaillées sur les erreurs. Par exemple, si vous cliquez sur le lien `<Name>`, MapForce marquera cet élément dans le volet **Sortie**.



2.3.2 Généreration de code

Le générateur de code est une fonction MapForce intégrée qui vous permet de générer du code depuis des fichiers de mappage. Vous pouvez utiliser le code généré pour exécuter vos mappages à l'extérieur de MapForce, qui vous permettra d'automatiser vos opérations de mappage. Vous pouvez générer du code dans les langages de transformation des données [17] suivants :

- XSLT 1.0/XSLT 2.0/XSLT 3.0 (*toutes éditions*)
- XQuery (*éditions Professional et Enterprise*)
- Java (*éditions Professional et Enterprise*)
- C# *éditions Professional et Enterprise*)
- C++ *éditions Professional et Enterprise*)

Vous pouvez générer du code depuis un seul design de mappage (**.mfd**) ou depuis un projet de mappage (**.mfp**). La génération de code depuis un projet est pris en charge uniquement dans les éditions Professional et Enterprise. Pour les détails, voir les sous-sections ci-dessous.

Points importants
Veuillez noter les aspects liés à la génération de code suivants :

- Certaines fonctions MapForce ne sont pas prises en charge dans du code de programme généré. Pour plus de détails, voir Fonctions prises en charge dans le code généré [484].
- Pour plus d'information concernant les chemins de handling dans le code généré, voir Chemins dans les environnements d'exécution [44].
- *Éditions Professional et Enterprise :* Vous pouvez changer les options générales de génération de code dans la section *Génération* du dialogue **Options** .
- *Éditions Professional et Enterprise :* La prise en charge pour les connexions de base de données varie selon dépendant de plateformes , et il existe des types de connexion qui ne sont pas pris en charge sur toutes les plateformes. Si votre mappage se connecte à une base de données, choisir une connexion de base de données qui est compatible avec l'environnement cible pour lequel vous générez du code.

Information relative à la prise en charge
La table ci-dessous résume l'information relative à la prise en charge de C++, C# et Java.

Langue cible	C++	C#	Java
Environnements de développement	Microsoft Visual Studio 2013, 2015, 2017, 2019, 2022	Microsoft Visual Studio 2013, 2015, 2017, 2019, 2022 Frameworks cible : • .NET Framework • .NET Core 3.1 • .NET 5.0 • .NET 6.0 • .NET 8.0	Java SE JDK 8, 11, 17, 21 (y compris OpenJDK) Eclipse 4.4 ou plus Apache Ant
mises en œuvre XML DOM	MSXML 6.0 Apache Xerces 3	System.Xml	JAXP
Database API	ADO	ADO.NET	JDBC

Générer du code depuis un mappage

Pour générer le code depuis un design de mappage (**.mfd**), suivez les instructions ci-dessous.

1. Sélectionnez les options de génération du code dans la section *Génération* du dialogue **Options** (applicable à C# et C++) dans le dialogue Paramètres de mappage 74.
2. Cliquez sur **Fichier | Générer code dans** et sélectionnez le langage de transformation pertinent. En alternative, vous pouvez sélectionner **Fichier | Générer le code dans le langage sélectionné**. Dans ce cas, le code sera généré dans le langage sélectionné dans la barre d'outils.
3. Sélectionnez un répertoire de destination pour les fichiers générés, puis cliquez sur **OK** pour confirmer. MapForce génère le code et affiche le résultat de l'opération dans la fenêtre Messages 25.

Le code généré d'un projet (éditions Professional et Enterprise)

Vous pouvez générer le code du projet de mappage (**.mfp**) qui consiste en de multiples fichiers de design de mappage (**.mfd**). Notez que les fichiers de design de mappage dans le projet doivent être qualifiés pour la génération, ce qui signifie que tous leurs composants doivent être pris en charge dans le langage de transformation sélectionné (voir Fonctions prises en charge dans le code généré 484).

Pour générer le code depuis un projet de mappage, suivez les instructions ci-dessous.

1. Ouvrir le projet de mappage pertinent pour lequel vous souhaitez générer le code.
2. Cliquer avec la touche de droite sur le nom du projet dans la fenêtre Projet et sélectionnez les **Propriétés** à partir du menu contextuel. En alternative, cliquez sur le nom du projet et sélectionnez l'item de menu **Projet | Propriétés**.
3. Revoir et changer les paramètres de projet, si requis. En particulier, assurez-vous que le langage cible et le répertoire de sortie sont définis correctement. Ensuite, cliquez sur **OK**.
4. Cliquez sur **Générer code pour tout le Projet** dans le menu **Projet**.

Indépendamment du langage sélectionné dans le dialogue **Propriétés de projet**, vous pouvez toujours choisir de générer le code de projet dans un langage différent, en sélectionnant la commande du menu **Projet | Générer code dans | <language>**.

Le progrès et le résultat du processus de génération de code est affiché dans la fenêtre Messages. Par défaut, le nom de l'application générée est le même que le nom du projet. Si le nom du projet contient des espaces, elles sont converties en traits de soulignement dans le code généré. Par défaut, le code est généré dans le même répertoire que le projet MapForce, dans le sous-répertoire **sortie**.

Vous pouvez changer le répertoire de sortie et/ou le nom du projet dans le dialogue **Propriétés de projet**. Si votre projet MapForce contient des dossiers, vous pouvez configurer les paramètres de génération de code pour chaque dossier individuel : Cliquez avec la touche de droite sur un dossier d'intérêt et sélectionnez **Propriétés** depuis le menu contextuel. Autrement, tous les dossiers de projet héritent des paramètres définis au niveau supérieur.

Étapes suivantes

Dépendant du langage de transformation que vous avez sélectionné pour la génération de code, les étapes suivantes varient. Si vous avez généré du code dans XSLT 1-3 ou XQuery, la prochaine étape sera d'exécuter la transformation depuis la ligne de commande (*voir les détails ci-dessous*).

Si vous avez généré Java, C#, ou le code C++, les prochaines étapes seraient de créer et d'exécuter le code généré. Vous pouvez aussi modifier le code Java, C# et C++ généré et l'intégrer dans votre code personnalisé.

Code XSLT et XQuery
Une fois que vous avez généré le code XSLT 1-3, le dossier de destination inclura les fichiers suivants :

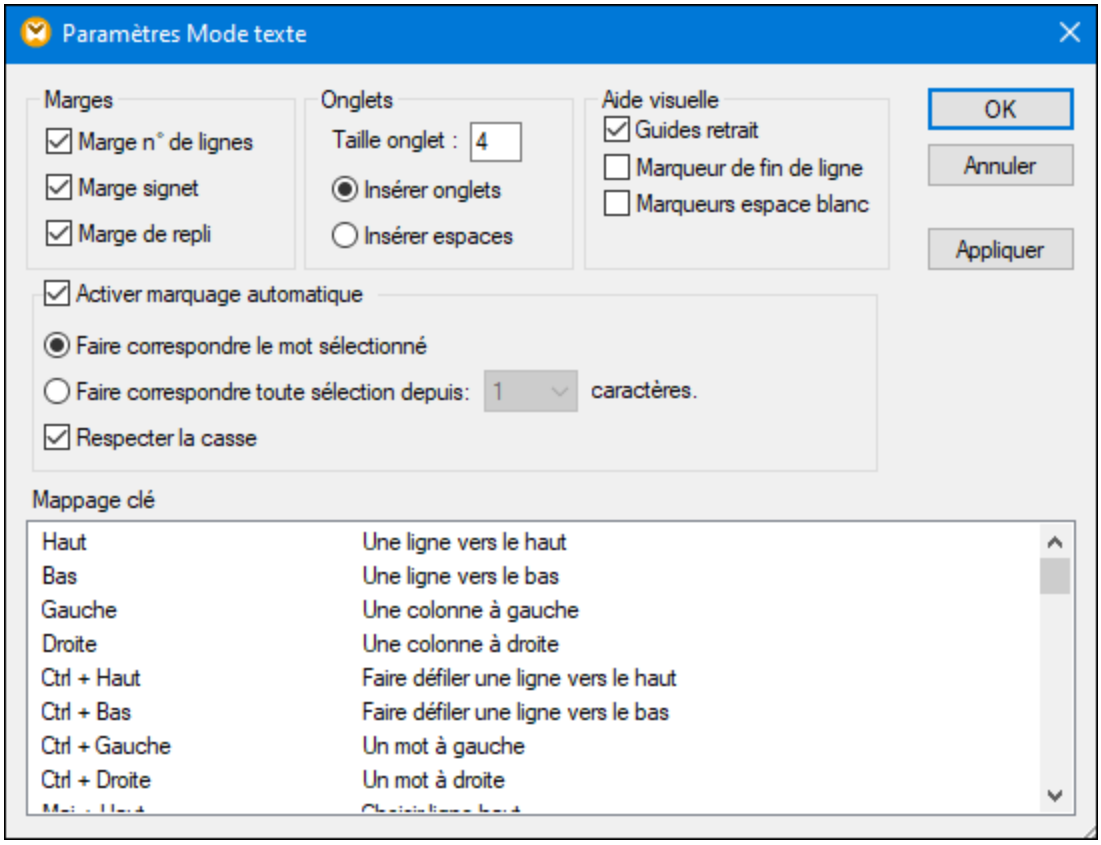
1. Le fichier XSLT transformation a le format suivant : **<Mapping>MapTo<TargetFileName>.xslt**. <Mapping> est la valeur du champ *Nom d'application* dans les paramètres de mappage[74]. <TargetFileName> est le nom du composant cible. Pour changer cette valeur, ouvrir les paramètres du composant cible et éditer la valeur du champ *Nom de composant*. Pour plus d'information, voir Modifier Paramètres de composant[39] et Chemins de bibliothèque dans du code généré[45].
2. Un fichier **DoTransform.bat**, qui vous permet d'exécuter la transformation XSLT avec Altova RaptorXML Server depuis la ligne de commande. Pour exécuter la commande, vous allez devoir installer RaptorXML.

Si votre mappage est chained[94], un fichier de transformation séparé sera généré pour chaque composant cible.

La génération de code XQuery est semblable à la génération de code XSLT, à l'exception du/des fichier/s de transformation qui ont l'extension **.xq** et le format suivant : **<Mapping>MapTo<TargetFileName>.xq**.

2.3.3 Fonctions de Mode Texte

Les volet de sortie[28] et volet XSLT[27] sont dotés de nombreuses aides visuelles afin de rendre l'affichage du texte plus facile : par ex., les marges, le marquage de texte, les guides de retrait, les marqueurs de fin de ligne et les marqueurs d'espace blanc. Vous pouvez personnaliser ces fonctions dans la boîte de dialogue **Paramètres du mode Texte** (*voir la capture d'écran ci-dessous*). Les paramètres dans le dialogue s'appliquent à l'application entière.



Pour ouvrir le dialogue **Paramètres Mode Texte**, suivez une des étapes suivantes :

- Sélectionnez **Sortie | Paramètres Mode Texte**.
- Cliquez sur (**Paramètres Mode Texte**) dans la barre d'outils.
- Cliquez avec la touche de droite sur la zone vide du volet **Sortie** et sélectionnez **Paramètres Mode texte** depuis le menu contextuel.

Certaines des aides à la navigation peuvent être activées depuis la barre d'outils **Mode Texte**, le menu d'application ou les raccourcis de clavier. Pour d'information sur les raccourcis, voir la section **Key Map** du dialogue des **Paramètres Mode Texte** affiché ci-dessus.

Voir la liste des paramètres disponibles ci-dessous.

- Marges

  Marge à lignes numérotées
  La numérotation des lignes est affichée dans la marge de numérotation, qui peut être activée ou désactivée dans le dialogue **Paramètres Mode Texte**. Lorsqu'une portion de texte est réduite, la portion de numérotation du texte réduit est également dissimulée.

  Marge à signet
  Les lignes dans le document peuvent être marquées par un signet pour une référence et un accès rapides. Si la case à cocher **marge à signet** dans le dialogue des **Paramètres Mode Texte** est

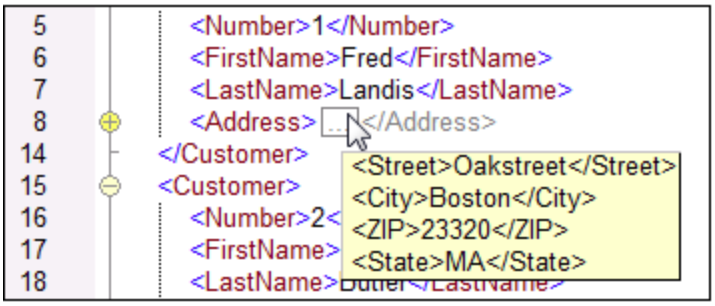
sélectionnée, les signets sont affichés dans la marge à signet (*voir la capture d'écran ci-dessous*). Si la case à cocher **marge à signet** n'est pas sélectionnée, les lignes marquées d'un signet de couleur cyan.

Vous pouvez éditer et parcourir les signets utilisant les commandes affichées dans la table ci-dessous. Les commandes sont disponibles dans le menu **Sortie** et également à travers le menu contextuel lorsque vous cliquez sur la touche droite du volet **Sortie**, **XSLT** ou **XQuery**.

	**Insérer/Supprimer Signet (Ctrl + F2)**
	**Aller au Signet suivant (F2)**
	**Aller au Signet précédent (Shift + F2)**
	**Supprimer tous les Signets (Ctrl + Shift + F2)**

Marge pliable

Le pliage de source se réfère à la capacité d'élargir et de réduire les nœuds. Cette fonction est affichée dans la marge pliable de source. La marge peut être activée et désactivée dans le dialogue **Paramètres Mode Texte**. Pour agrandir ou réduire des portions de texte, cliquer sur les nœuds + et - du côté gauche de la fenêtre. Toute portion du code réduit est affichée avec un symbole d'ellipse(*voir la capture d'écran ci-dessous*). Pour consulter le code réduit sans l'agrandir, déplacez le curseur de la souris sur l'ellipse. Une infobulle s'ouvre qui affiche le code à consulter, tel qu'indiqué dans la capture d'écran ci-dessous. Veuillez noter que si le texte consulté est trop grand pour être affiché dans l'infobulle, une ellipse supplémentaire apparaît à la fin de l'infobulle.



- Activer le marquage automatique

  Le paramètre **Activer le marquage automatique** vous permet de voir toutes les correspondances du morceau de texte sélectionné. La sélection est marquée en bleu clair, et les correspondances sont marquées en brun clair. La sélection et ses correspondances sont marquées en tant que carrés gris dans la barre de défilement. La position actuelle du curseur est affichée en tant que marqueur bleu dans la barre de défilement. Une sélection peut être définie comme étant un mot entier ou un nombre fixe de caractères. Vous pouvez aussi spécifier si la casse doit être prise en compte ou pas.

  Pour sélection de caractère, vous pouvez spécifier le nombre minimum de caractères qui doivent correspondre, en commençant depuis le premier caractère dans la sélection. Par exemple, vous pouvez choisir de faire correspondre deux caractères ou plus. En ce qui concerne les recherches de mots, les éléments suivants sont considérés être des mots séparés : noms d'élément (sans crochets), les crochets de balises d'éléments, les noms d'attributs et les valeurs d'attribut sans guillemets.

- Aide visuelle

  Guides de retrait
  Les guides de retrait sont des lignes verticales qui indiquent l'étendue d'un retrait d'une ligne. Ils peuvent être activés et désactivés dans le dialogue **Paramètres Mode Texte**. Les options **Insérer onglets** et **Insérer espaces** prennent effet quand vous utilisez l'option **Sortie | Texte XML Pretty-Print**.

  Marqueurs de fin de ligne et marqueurs d'espace blanc
  Les marqueurs de fin de ligne et les marqueurs d'espace blanc (*voir la capture d'écran ci-dessous*) peuvent être basculés dans le dialogue **Paramètres Mode Texte**. Les flèches représentent les tabulations. L'abréviation *CR* est synonyme de retour de chariot. Les pointillés représentent les espaces de caractères.

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

- Autres paramètres du Mode Texte

  Coloration syntaxique
  La coloration syntaxique est une autre aide visuelle qui rendent la lecture des listes de codes plus conviviale. La coloration syntaxique dépend de la valeur sémantique du texte. Par exemple, dans des documents XML, selon que le nœud XML est un élément, un attribut, un contenu, une section CDATA, un commentaire ou une instruction de traitement, le nom de nœud (et dans certains cas, le contenu du nœud) sera coloré d'une autre teinte.

  Zoom avant et arrière
  Vous pouvez zoomer en avant et en arrière (en faisant défiler la roulette de la souris) tout en maintenant la touche **Ctrl** appuyée. En alternative, appuyez sur les touches - ou + tout en maintenant la touche **Ctrl** appuyée.

Pretty-print
La commande **Pretty-Print XML Text** reformate le document XML actif dans le **Mode Texte** pour donner un affichage structuré du document. Par défaut, chaque nœud enfant est séparé de son parent par quatre espaces. Cela peut être personnalisé depuis le dialogue **Paramètres Mode Texte**. Pour imprimer automatiquement un document XML, sélectionnez la commande de menu **Sortie | Pretty-Print XML Text** ou cliquez sur (**Pretty-print**) dans la barre d'outils.

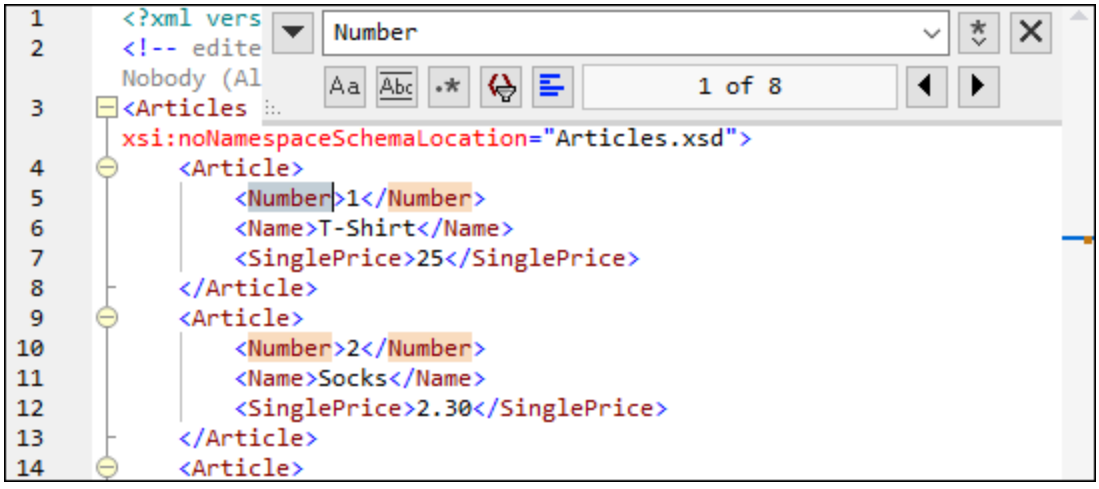
Retour automatique à la ligne
Le retour à la ligne aide à afficher une liste de codes à l'intérieur des bords de l'espace de travail. Si le paramètre de retour à la ligne n'est pas activé, certaines parties de texte peuvent ne pas être entièrement visibles dans l'espace de travail. Pour activer le retour automatique à la ligne dans le document actif, sélectionnez la commande de menu **Sortie | Word Wrap** ou cliquez sur (**Word Wrap**) dans la barre d'outils.

2.3.4 Recherche Mode Texte

Le texte dans le volet **Sortie**, et le volet **XSLT** peut être recherché avec un plage d'options et d'aides visuelles extensives.

Vous pouvez rechercher un terme dans tout le document ou à l'intérieur d'une sélection de texte. Pour démarrer une recherche , appuyez sur **Ctrl+F** ou sélectionnez la commande de menu **Édition | Recherche**. Vous pouvez saisir un string ou utiliser la zone de liste déroulante pour sélectionner un string d'un des derniers 10 strings. Quand vous saisissez ou sélectionnez un string, toutes les correspondances sont mises en surbrillance, et les positions des correspondances sont indiquées par des marqueurs orange dans la barre de défilement (*voir la capture d'écran ci-dessous*). La position de la correspondance sélectionnée actuellement (en surbrillance grise) dépend du dernier emplacement du curseur.

Vous pouvez voir le nombre total de correspondances et la position de l'index de la correspondance actuelle sélectionnée. Utilisez les boutons (**Précédent**) et (**Suivant**) pour basculer entre les correspondances.



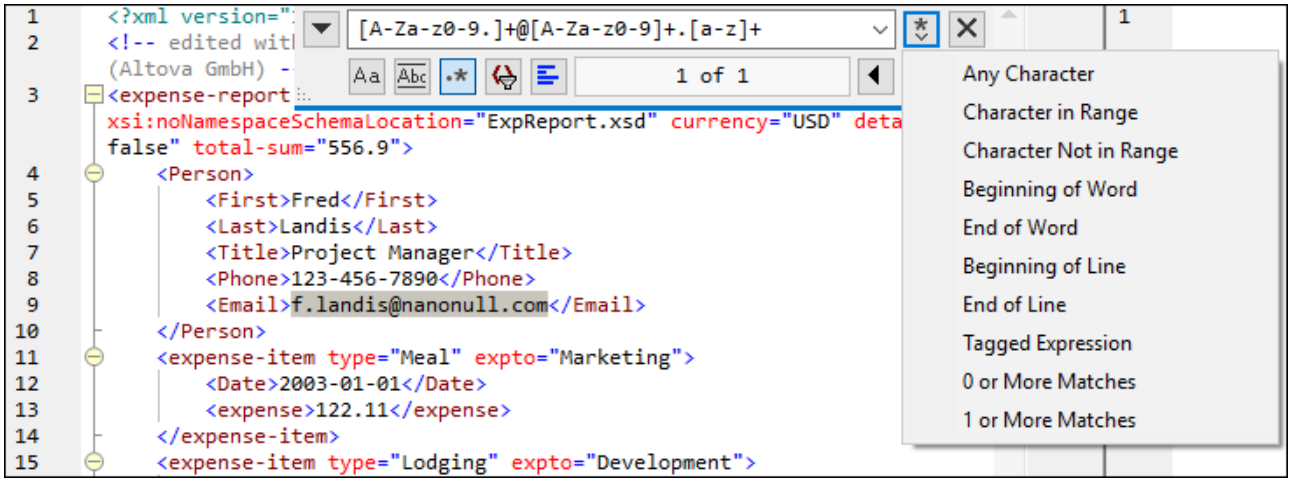
Options de Recherche

Vous pouvez spécifier des critères de recherche à l'aide des boutons situés sous le champ de recherche. La liste des options disponibles est donnée dans la table ci-dessous.

Option	Icône	Description
Respecter la casse	Aa	Réalise une recherche sensible à la casse : par ex., *Address* n'est pas la même chose que *address*.
Correspondance du mot entier	Abc	Seuls les mots identiques correspondront.
Utiliser une expression régulière	.*	Une fois activé, le terme de recherche sera lu en tant qu'expression régulière. Voir les *expressions régulières* ci-dessous.
Trouver l'ancre		La position de l'ancre dépend du dernier emplacement du curseur. Cliquez sur les boutons **Précédent** et **Suivant** ne modifie pas la position de l'ancre.
Trouver dans la sélection		Une sélection est un morceau de texte marqué. Pour trouver un terme au sein d'une sélection, marquez un morceau de texte, appuyez sur **Ctrl+F**, veillez à ce que le bouton **Chercher dans la sélection** est appuyé et saisissez le terme dans le champ de recherche.

Expressions régulières

Vous pouvez utiliser des expressions régulières (regex) pour trouver un string de texte. Pour ce faire, activez l'option **Utiliser les expressions régulières** (*voir la table ci-dessus*). Ensuite, saisissez une expression régulière dans le champ de recherche Cliquer sur (**Générateur d'expression régulière**) vous donne une liste d'exemples d'expression régulière (*voir ci-dessous*). La capture d'écran ci-dessous montre une expression régulière qui aide à trouver des adresses e-mail.



Métacaractères d'expression régulière
La table ci-dessous affiche des métacaractères que vous pouvez utiliser, et remplacer du texte. Tous les métacaractères à l'exception des deux derniers correspondent aux éléments de menu dans le **Générateur d'expression régulière** (*voir ci-dessus*).

Item de menu	Métacaractères	Description
Tout caractère	`.`	Correspond à n'importe quel caractère. Il s'agit d'un espace réservé pour un seul caractère.
Gammes de caractère	`[...]`	Correspond à tout caractère se trouvant dans cet ensemble. Par exemple, **`[abc]`** correspond à tous les caractères `a`, `b` ou `c`. Vous pouvez aussi utiliser des plages : par ex., **`[a-z]`** pour tout caractère en minuscule.
Caractères absent de la plage de caractères	`[^...]`	Correspond à tout caractère ne se trouvant pas dans cet ensemble. Par exemple, **`[^A-Za-z]`** correspond à tout caractère sauf à un caractère alphabétique.
Début du mot	`\<`	Correspond au début d'un mot.
Fin du mot	`\>`	Correspond à la fin d'un mot.
Début de ligne	`^`	Correspond au début d'une ligne sauf si utilisé à l'intérieur d'un ensemble(*voir ci-dessus*).
Fin de ligne	`$`	Correspond à la fin d'une ligne. Par exemple, **`A+$`** correspond à un ou plusieurs `A` à la fin de la ligne.
Expression balisée	`(abc)`	Les parenthèses marquent le début et la fin d'une expression balisée. Les expressions balisées peuvent être utiles lorsque vous devez baliser (« vous souvenir ») d'une région comparée afin de pouvoir vous y référer ultérieurement. Il est possible de baliser jusqu'à neuf sous-expressions, puis de les rétro-référencer ultérieurement. Par exemple, **`(the) \1`** correspond au string `the the`. Cette expression peut être expliquée comme suit : Faites correspondre le string `the` et souvenez-vous en comme région balisée ; l'expression doit être suivie d'un espace et d'une rétro-référence à la région balisée correspondante précédente.
0 ou plus de correspondances	`*`	Correspond à zéro ou plusieurs matches de l'expression précédente. Par exemple, **`Sa*m`** correspond à `Sm`, `Sam`, `Saam`, `Saaam` etc.
1 ou plus de correspondances	`+`	Correspond à une ou plusieurs occurrences de l'expression précédente. Par exemple, **`Sa+m`** correspond à `Sam`, `Saam`, `Saaam` etc.
	`\n`	Où `n` est `1` à `9` , `n` se réfère à la première région balisée de neuf régions (*voir ci-dessus*).
	`\x`	Vous permet d'utiliser un caractère `x`, qui aurait généralement une signification particulière. Par exemple, `\[` sera interprété comme `[` et non pas comme le début d'un jeu de caractères.

Chercher caractères spéciaux

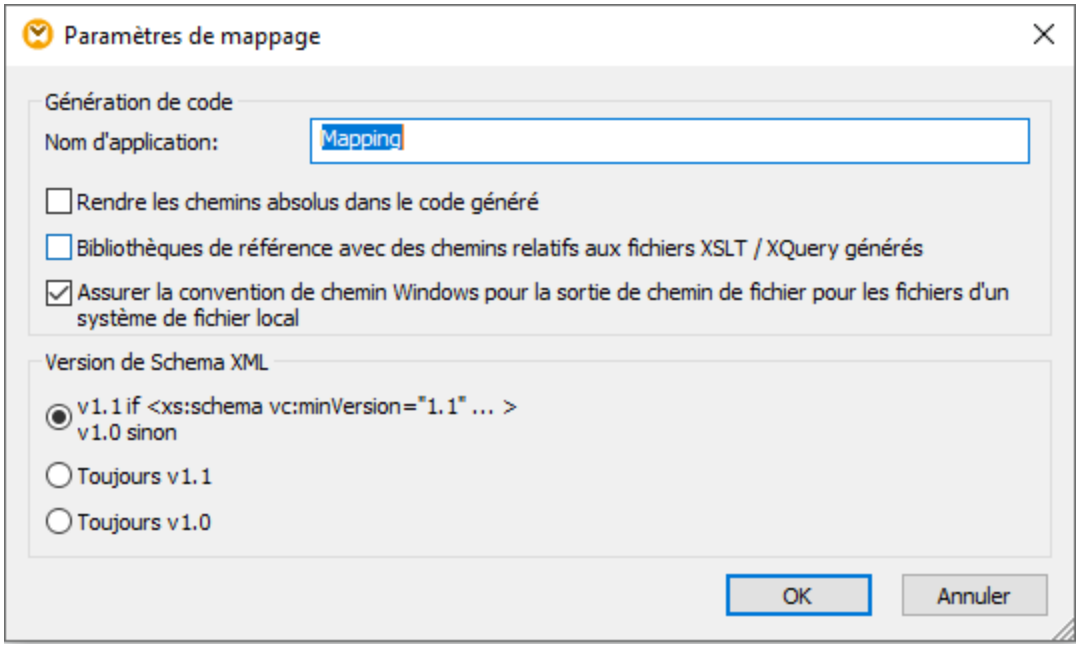
Si l'option **Utiliser des expressions régulières** est activée, vous pouvez rechercher n'importe quel des caractères spéciaux suivants au sein du texte :

- `\t` (Onglet)
- `\r` (Retour de chariot)
- `\n` (Nouvelle ligne)
- `\\` (Backslash)

Par exemple, pour trouver un caractère de tabulation, appuyez sur **Ctrl + F**, choisissez l'option **Utiliser les expressions régulières**, puis saisissez `\t` dans le dialogue **Recherche**.

2.3.5 Paramètres de mappage

La boîte de dialogue **Paramètres de mappage** (*voir la capture d'écran ci-dessous*) vous permet de définir les paramètres spécifiques au document. Pour ouvrir cette boîte de dialogue, allez au menu **Fichier** et cliquez sur **Paramètres de mappage**. En alternative, cliquez avec la touche de droite dans la zone vide du volet de mappage et sélectionnez **Paramètres de mappage** depuis le menu contextuel.



Les paramètres disponibles sont décrits dans les sous-pages ci-dessous.

- Génération de code
  - *Nom d'application :* Définit le préfixe du fichier XSLT généré ou le nom d'application Java, C# ou C++ (*éditions Professional et Enterprise*).

- *Nom de package de base Java (éditions Professional et Enterprise) :* Cette option s'applique lorsque Java est sélectionné en tant que langage de transformation. L'option définit le nom de package de base pour la sortie Java.

- Rendre les chemins absolus dans le code généré : Cette case à cocher touche tous les chemins dans les composants de mappage, sauf les chemins vers les fichiers de bibliothèque externes (comme les bibliothèques XSLT). La case à cocher définit si les chemins de fichier doivent être relatifs ou absolus dans le code de programme généré. Pour plus d'informations, voir Chemins dans des environnements d'exécution [44].

- *Bibliothèques de référence avec des chemins relatifs aux fichiers XSLT/XQuery générés :* Cette case à cocher s'applique quand le langage de mappage est XQuery (*éditions Professional et Enterprise*), soit XSLT. Cette option est utile si votre mappage référence une bibliothèque XSLT ou XQuery et si vous prévoyez générer des fichiers XSLT ou XQuery depuis le mappage. Sélectionnez cette case à cocher si vous souhaitez que les chemins de bibliothèque soient relatifs au répertoire du code XSLT ou XQuery généré. Si la case à cocher n'est pas sélectionnée, les chemins de bibliothèque seront absolus dans le code généré. Voir aussi chemins de Bibliothèque dans le code généré [45].

- *Assurer la convention de chemin Windows pour le chemin de fichier :* Cette case à cocher s'applique si le langage de mappage est XQuery (*éditions MapForce Professional et Enterprise*), XSLT 2.0 ou XSLT 3.0. Cette case à cocher garantit que les conventions de chemin Windows sont suivies. Lors de la sortie XSLT 2.0, XSLT 3.0 ou XQuery, le nom de fichier traité actuellement est extrait en interne et en avec l'aide de la fonction **`document-uri`**, qui retourne un chemin sous le format `file://URI` pour des fichiers locaux. Lorsque cette case à cocher est sélectionnée, un chemin de spécification `file://URI` est converti automatiquement dans un chemin de fichier Windows complet (e.g., `C:\...`) pour simplifier le traitement supplémentaire.

Paramètres de fichier de sortie (éditions Professional et Enterprise)

La zone de liste **fin de ligne** vous permet de spécifier les fins de ligne des fichiers de sortie. La *Plateforme par défaut* signifie l'option par défaut pour les systèmes d'exploitation cible : par exemple, Windows (CR+LF), macOS (LF) ou Linux (LF). Vous pouvez aussi choisir une fin de ligne spécifique manuellement. Les paramètres que vous choisissez ici sont importants lorsque vous compilez un mappage vers un fichier d'exécution de MapForce Server (`.mfx`) ou lorsque vous déployez un mappage sur FlowForce Server exécuté sur un système d'exploitation différent.

Version de Schéma XML

Cette option permet de définir la version de schéma XML utilisée dans le fichier de mappage. Veuillez noter que pas toutes les versions 1.1. spécifiques sont actuellement prises en charge. Si la déclaration `xs:schema vc:minVersion="1.1"` est présente, la version 1.1 sera utilisée ; sinon, la version 1.0 sera utilisée.

Si le document XSD n'a pas d'attribut `vc:minVersion` ou si la valeur de l'attribut `vc:minVersion` est autre que `1.0` ou `1.1`, alors XSD 1.0 sera le mode par défaut. Ne confondez pas l'attribut `vc:minVersion` avec l'attribut `xsd:version`. Le premier attribut a le numéro de version XSD, alors que le deuxième contient le numéro de version du document . Changer ce paramètre dans un mappage existant entraîne un rechargement de tous les schémas de la version de schéma XML sélectionnée, et peut aussi modifier sa validité.

Paramètres d'opération de service Web (édition Enterprise)

Les champs de **WSDL Definitions**, **Service**, **Point de terminaison** et **Opération** sont automatiquement remplis si le document de mappage fait partie de la mise en place de service Web.

3 Tutoriels

Avec l'aide des tutoriels, vous serez à même de comprendre et utiliser les capacités de transformation de données basiques de MapForce. Les tutoriels vous guideront à travers les bases et ce étape par étape. Au fur et à mesure, les tutoriels deviennent de plus en plus complexes. Pour cette raison, il est recommandé de les suivre par séquence. Il est avantageux de posséder des connaissances de XML et de Schéma XML.

Exemples de fichiers
Les fichiers de mappage illustrés ou référencés dans ces tutoriels sont disponibles dans le dossier BasicTutorials[16]. Si vous avez des doutes sur les possibles effets qu'aura une modification sur les exemples originaux fournis avec MapForce, pensez à créer des sauvegardes avant de procéder aux modifications.

Liste des tutoriels

Une source vers une cible
Ce tutoriel[78] montre comment utiliser les mécanismes clés de MapForce pour mapper les nœuds d'un fichier source aux nœuds d'un fichier cible. Ce tutoriel poursuit son explication comment convertir un fichier XML défini par un schéma XML en fichier XML défini par un schéma XML différent.
Sources multiples vers une cible
Ce tutoriel[89] montre comment fusionner des données depuis de multiples fichiers XML source à un fichier cible.
Mappages enchaînés
Dans ce tutoriel[94], nous créons un mappage simple comme dans le second tutoriel, puis nous filtrons les données produites par ce mappage et passons les données filtrées au deuxième fichier cible.
Sources multiples vers cibles multiples
Ce tutoriel[102] montre comment lire des données depuis multiples fichiers d'instance XML situés dans le même dossier et de les écrire dans de multiples fichiers XML générés instantanément.

3.1 Une source vers une cible

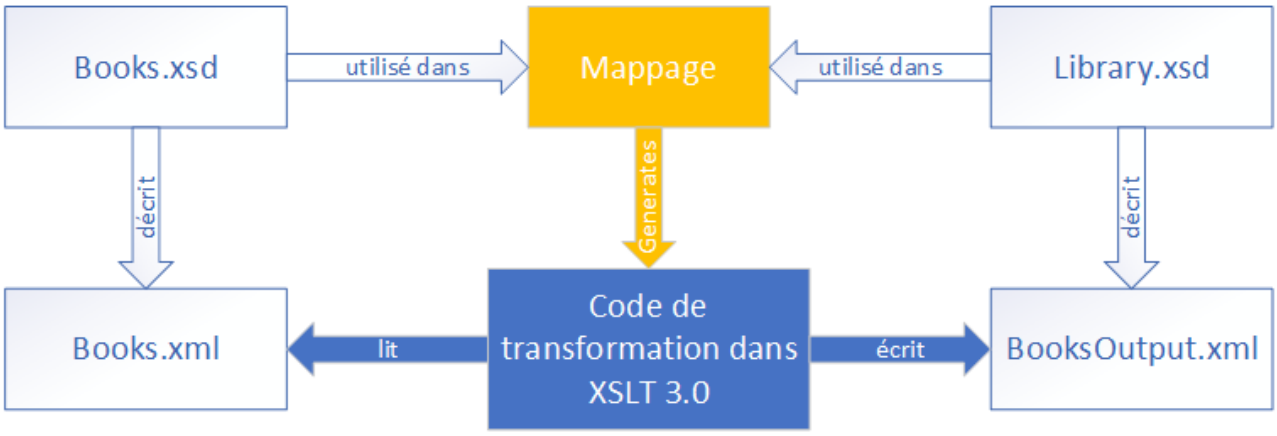
Ce tutoriel décrit comment créer un mappage pour un des scénarios les plus basiques. Notre objectif est de transférer les données depuis un fichier XML avec un schéma XML A y assigné) et mettre ces données dans un fichier XML B (avec un schéma XML B y assigné). Les grandes lignes de notre méthode seront comme suit :

1. Puisque nous utilisons deux structures de données, nous allons créer deux composants (*Source* et *Cible*) dans notre conception de mappage.
2. Puisque la transformation d'un document en un autre est réalisée en utilisant un langage de transformation approprié langage de transformation[17], nous sélectionnons un langage de transformation.
3. Puis, nous allons mapper des nœuds en connectant un nœud source à un nœud cible désiré. Ce sont ces connexions qui constituent le mappage et déterminent quel nœud source correspond à quel nœud cible.
4. Le résultat du mappage nous permettra d'obtenir le document XML cible qui est valide conformément au schéma cible.
5. Finalement, nous allons pouvoir enregistrer le fichier XML.

Pour plus d'information sur la manière dont la transformation de données est effectuée, voir le modèle abstrait ci-dessous.

Modèle abstrait

Le modèle abstrait ci-dessous illustre la transformation de données dans ce tutoriel :



Notre mappage a une source et une cible. Le schéma source (**Books.xsd**) décrit la structure du fichier de l'instance source (**Books.xml**). Le schéma cible (**Library.xsd**) décrit la structure du fichier de l'instance cible (**BooksOutput.xml**). Lorsque vous connectez les nœuds de source aux nœuds correspondants de la cible, le mappage génère le code de transformation dans XSLT 2.0. Le code de transformation lit les données de **Books.xml** et écrit ces données dans **BooksOutput.xml**.

Fichiers source et cible

La liste de codes ci-dessous montre les données échantillon de **Books.xml** qui seront utilisées en tant que source de données.

```
<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>
```

C'est ainsi que nous voulons que nos données aient l'air dans le fichier cible dénommé **BooksOutput.xml** :

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

Certains noms d'élément dans l'XML source et cible ne sont pas identiques. Notre objectif est de remplir les éléments `<author>`, `<title>`, `<genre>` et `<publish_year>` du fichier cible depuis les éléments équivalents dans le fichier source (`<author>`, `<title>`, `<category>`, `<year>`). L'attribut `id` dans le fichier source doit être mappé vers l'élément `<id>` dans le fichier cible. Enfin, nous devons remplir l'élément `<last_updated>` du fichier cible avec la date et l'heure à laquelle le fichier a été mis à jour dernièrement.

Pour effectuer la transformation de données requise, suivez les étapes décrites dans les sous-sections ci-dessous.

3.1.1 Créer et enregistrer le Design

Cette rubrique explique comment créer un nouveau design, sélectionner un langage de transformation, valider et enregistrer votre mappage.

Créer un nouveau design

Afin de pouvoir effectuer une transformation, vous devrez créer un nouveau design de mappage, qui peut être fait d'une des manières suivantes :

- Allez au menu **Fichier** and cliquez sur **Nouveau**.
- Cliquez sur dans la batte d'outils.

Sélectionner un langage de transformation

Dépendant de votre édition de MapForce, différents langages de transformation[17] sont disponibles. Pour ce tutoriel, nous avons choisi XSLT3. Vous pouvez sélectionner ce langage de transformation d'une des manières suivantes :

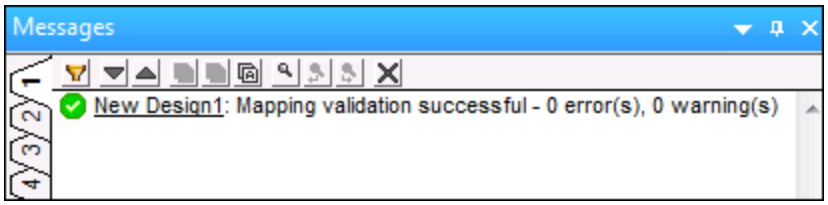
- Cliquez sur **XSLT3** dans la barre d'outils.
- Ouvrez le menu **Sortie** et cliquez sur **XSLT 3.0**.

Valider et enregistrer le mappage

La validation d'un mappage est une étape optionnelle qui vous permet de voir et de corriger des erreurs et des avertissements de mappage potentiels avant d'exécuter le mappage. Vous pouvez valider votre mappage à toute étape. Pour vérifier si le mappage est valide, effectuez une des étapes suivantes :

- Cliquez sur **Valider mappage** dans le menu **Fichier**.
- Cliquez sur dans la batte d'outils.

La fenêtre Messages affiche les résultats de validation comme suit :



Pour enregistrer le mappage, effectuez une des étapes suivantes :

- Cliquez sur **Enregistrer** dans le menu **Fichier**.
- Cliquez sur dans la batte d'outils.

À votre convenance, le mappage créé dans ce tutoriel est enregistré en tant que **`Tut1_OneToOne.mfd`**.

3.1.2 Ajouter composant source

À cette étape, nous souhaitons ajouter un fichier XSD file, qui sera la structure du premier composant, et un fichier XML, qui fournira les données pour ce composant. Le fichier source dénommé **`Books.xsd`** peut être ajouté au mappage d'une des manières suivantes :

- Cliquez  (**Insérer Fichier/Schéma XML**) dans la barre d'outils.
- Dans le menu **Insérer**, cliquez sur **Schéma XML/Fichier**.
- Glissez **Books.xsd** depuis Windows Explorer dans la zone de mappage.

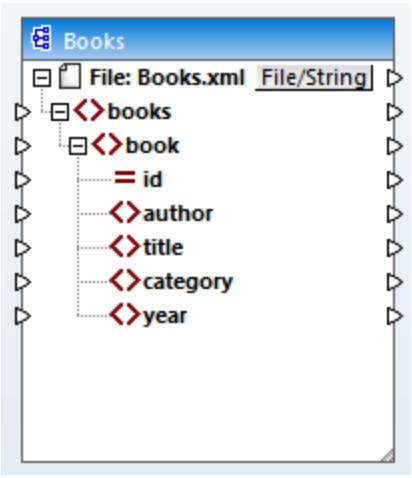
Si vous sélectionnez une des deux premières options, le dialogue **Insérer un fichier Schéma XML** proposera un choix entre un schéma pré-packagé ou un fichier local ou à distance. Dans nos tutoriels, tous les fichiers sont en local. Pour plus d'information sur l'ajout de fichiers XML, voir XML et Schéma XML 112.

Si un composant est créé depuis un fichier XSD, vous êtes invité à avoir un fichier XML qui sera utilisé en tant que fichier des données du composant. Si un composant est créé depuis un fichier XML, le fichier XSD qui est référencé depuis un fichier XML sera utilisé pour définir la structure des données du composant. Si aucune référence à un fichier XSD n'existe, MapForce proposera de générer un fichier XSD pour ce composant.

Puisque nous ajoutons d'abord un fichier de schéma, MapForce propose d'ajouter un exemple de fichier XML. Cliquez sur **Naviguer** et recherchez **Books.xml** qui est placé dans le même dossier. Donc notre fichier source contient un schéma et du contenu.

Afficher la structure

Maintenant que le fichier source a été ajouté à la zone de mappage, vous pouvez voir sa structure. Dans MapForce, cette structure est connue en tant qu'un composant de mappage ou simplement comme composant 30. Vous pouvez élargir des éléments dans le composant en cliquant sur l'icône ⊞ . De manière alternative, vous pouvez appuyer sur la clé **+** sur le pavé numérique. La capture d'écran ci-dessous illustre le composant source :



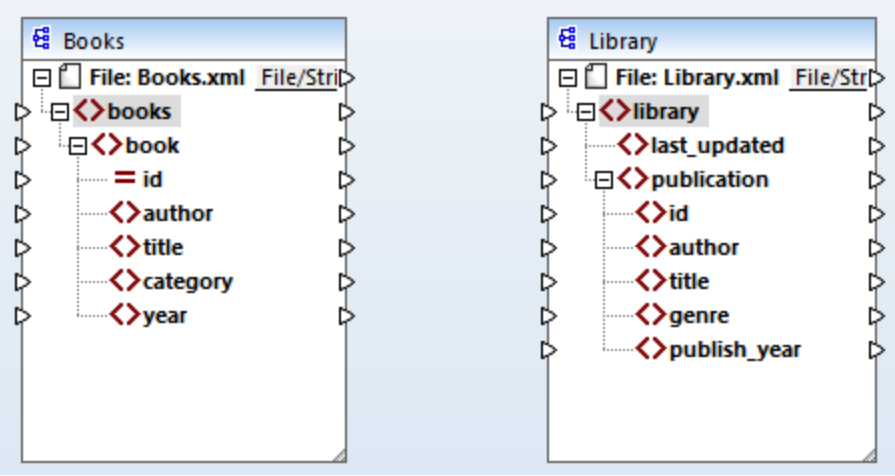
Le nom de l'en-tête (Books) fait uniquement référence au nom de composant et non pas au nom de schéma sur lequel ce fichier est basé. Pour voir le nom du schéma et d'autres propriétés du composant, double-cliquez sur l'en-tête du composant. Ceci ouvrira la boîte de dialogue Paramètres de composant 113.

Le nœud au niveau supérieur du composant (**Fichier : Books.xml**) représente le nom du fichier d'instance XML. Les éléments XML dans la structure sont représentés par l'icône  . Les attributs XML sont représentés par l'icône  . Les petits triangles, affichés des deux côtés du composant, représentent les entrées de données du côté gauche et les sorties du côté droit. Dans MapForce, ces triangles sont appelés *connecteurs d'entrée* et *connecteurs de sortie*, respectivement.

Pour plus d'information sur les composants, connexions, procédures générales et fonctions, voir Notions fondamentales de mappage 30.

3.1.3 Ajouter composant cible

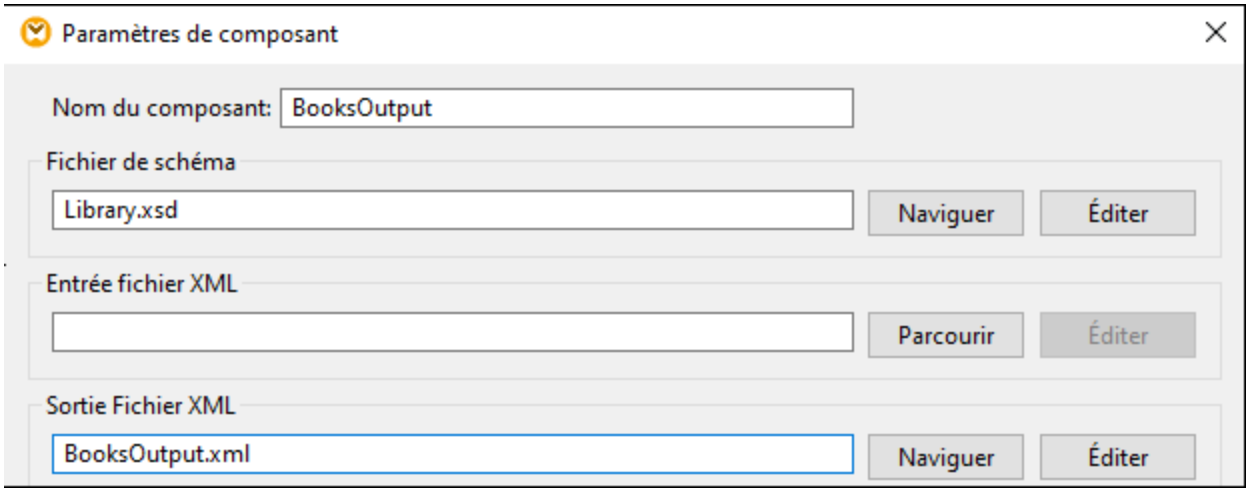
La prochaine étape est d'ajouter un composant cible et de définir ses paramètres. Ajoutez le fichier cible **`Library.xsd`** au mappage. Cliquez sur **Ignorer** lorsque vous êtes invité par MapForce à fournir un fichier d'instance. À cette étape, le design de mappage ressemble à l'exemple suivant :



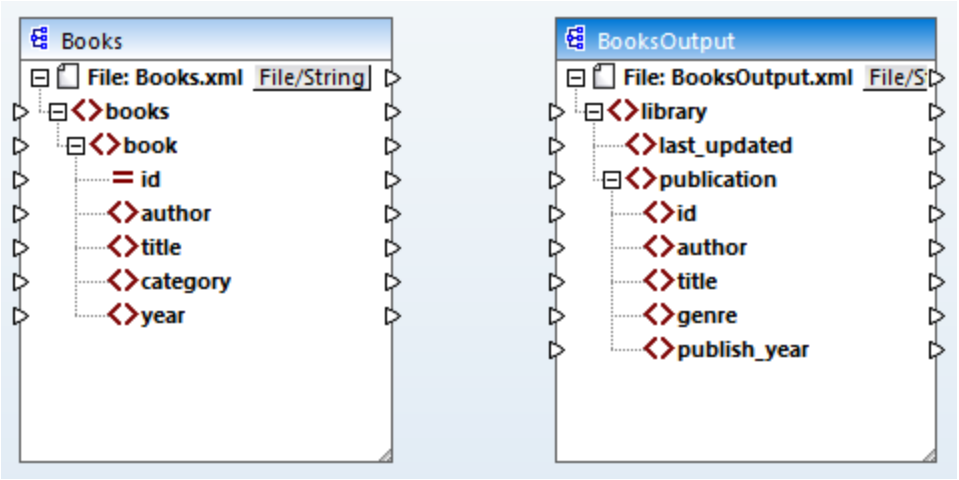
Veuillez noter que lorsque vous ouvrez **`library.xsd`**, il est affiché en tant que fichier XML file dans le composant. En fait, MapForce crée uniquement une référence vers le fichier XML dénommé **`Library.xml`**, mais ce fichier XML lui-même n'existe pas encore. Donc notre compostant cible a un schéma mais pas de contenu.

Paramètres de composant

Désormais, nous allons devoir renommer le composant cible **`BooksOutput`**. Le fichier de sortie sera appelé **`BooksOutput.xml`**. Ceci nous permettra d'éviter toute confusion dans les prochains tutoriels, étant donné que nous allons utiliser un fichier séparé dénommé **`Library.xml`**, qui a son propre contenu et est basé sur le même schéma **`Library.xsd`**. Pour renommer le fichier cible, double-cliquez sur l'en-tête du composant cible. Ceci ouvre la boîte de dialogue des Paramètres de composant 113 (*voir la capture d'écran ci-dessous*), dans laquelle nous devons changer les noms du composant cible et du fichier cible somme suit :



Le design de mappage a désormais l'air de ceci :



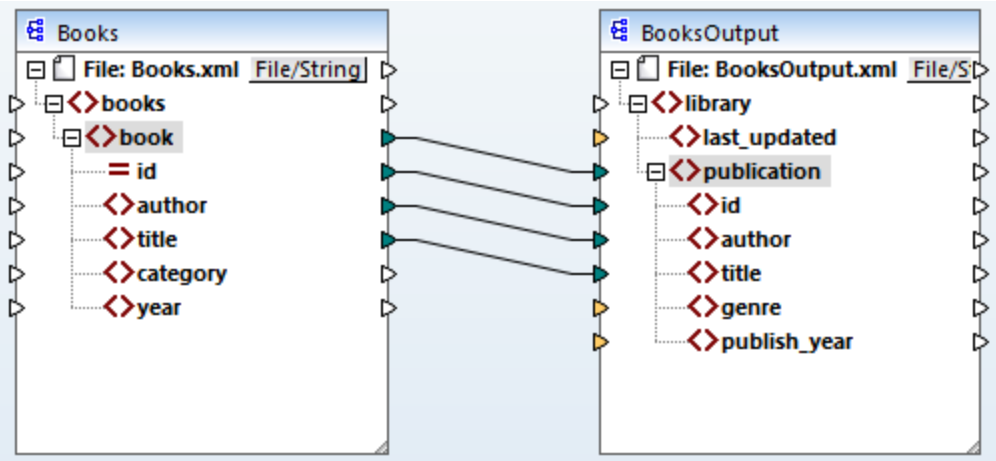
3.1.4 Connecter Source et Cible

À cette étape, nous allons mapper les données dans le fichier source au fichier cible. Nous fournirons également des informations sur la date et l'heure actuelles utilisant la fonction XPath `current-dateTime`[323].

Connexions automatiques

Désormais, nous allons créer une connexion de mappage entre l'élément <book> dans le composant source et l'élément <publication> dans le composant cible. À cette fin, appuyez et restez sur le connecteur de sortie (le petit triangle) situé à droite de l'élément <book> et glissez-le dans le connecteur d'entrée de l'élément <publication> dans la cible. Lorsque vous effectuez cette étape, MapForce peut connecter automatiquement tous les nœuds enfant de <book> dans le fichier source vers les éléments portant les mêmes noms dans le fichier cible. Dans notre exemple, quatre connexions ont été créées simultanément (*voir capture d'écran ci-*

dessous). Cette fonction est appelée *Auto-connexion des enfants correspondants* [52] et peut être désactivée et personnalisée, le cas échéant.



Vous pouvez activer ou désactiver **Auto-connexion des enfants correspondants** d'une des manières suivantes :

- Cliquez sur (**Basculer auto-connexion des enfants**) dans la barre de menu.
- Dans le menu **Connexion**, cliquez sur **Auto-connexion des enfants correspondants**.

Connecter des items obligatoires

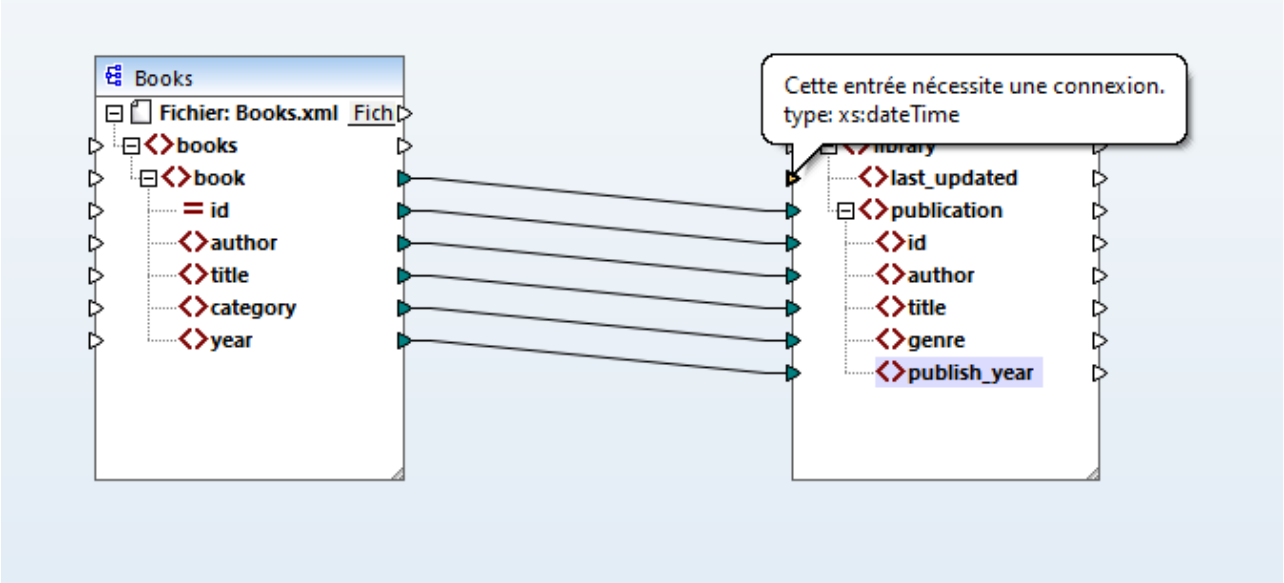
Veuillez noter que certains des connecteurs d'entrée dans le composant cible ont été marqués en orange par MapForce, ce qui indique que ces items sont obligatoires. Ces items sont obligatoires car ils ont été définis de cette manière dans le schéma de fichier. Afin de garantir la validité du fichier XML cible, il est nécessaire de fournir les valeurs pour les items obligatoires comme suit :

- Connectez l'élément `<category>` dans la source avec l'élément `<genre>` dans le composant cible.
- Connectez l'élément `<year>` dans la source avec l'élément `Lignes` dans le composant cible.

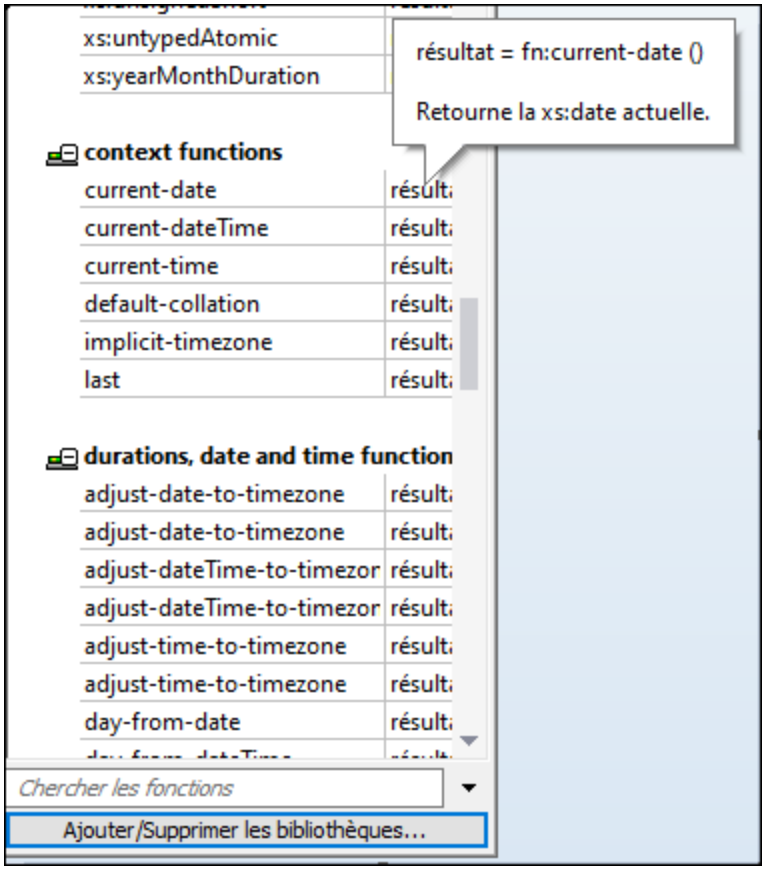
Ajouter la date et l'heure actuelles

Enfin, vous devez fournir une valeur pour l'élément `<last_updated>`. Si vous déplacez la souris sur son connecteur d'entrée, vous constaterez que l'élément est de type `xs:dateTime`(*voir capture d'écran ci-dessous*).

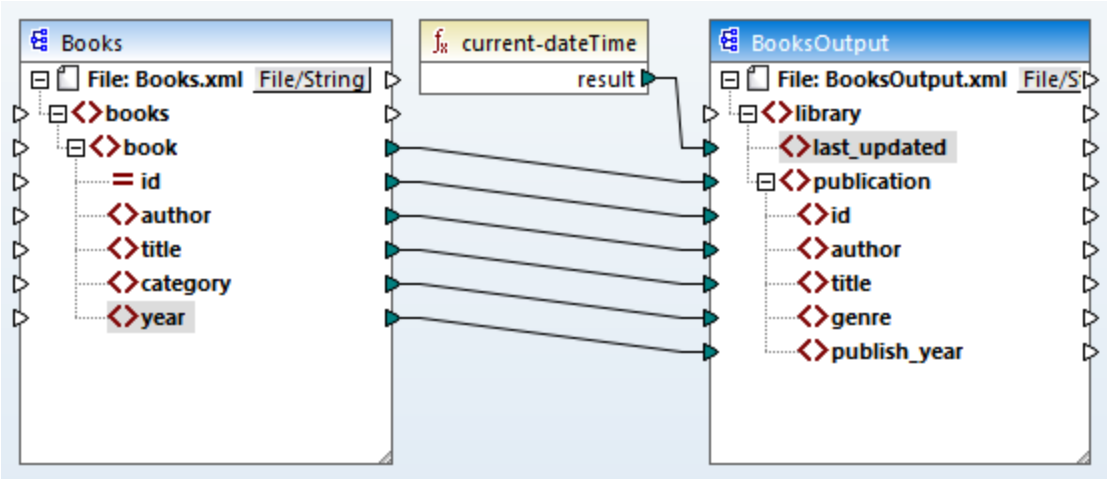
Pour voir les conseils, appuyez sur le bouton de la barre d'outils . En cliquant (**Afficher types de données**) dans la barre d'outils, vous pouvez aussi rendre le type de données visible à tout moment.



Vous pouvez recevoir la date et l'heure actuelles par le biais de la fonction **`current-dateTime`**. Pour trouver cette fonction, tapez dans la zone de texte située dans le bas de la fenêtre des bilbiothèques[21] (*voir la capture d'écran ci-dessous*). En alternative, double-cliquez sur une zone vide à l'intérieur du volet de mappage et commencez à saisir `current-date`.



Pour ajouter la fonction au-dessus du mappage, glissez-la depuis la fonction vers le volet de mappage. Puis connectez sa sortie dans l'entrée de l'élément `<last_updated>` (*voir la capture d'écran ci-dessous*).



Vous pouvez désormais valider et enregistrer votre mappage, tel qu'affiché dans Créer et enregistrer le design 79.

3.1.5 Consulter le résultat de mappage

MapForce utilise ses moteurs built-in pour générer la sortie et permet de prévisualiser le résultat du mappage directement dans le volet Sortie (*voir la capture d'écran ci-dessous*).

Enregistrer sortie
Par défaut, les fichiers affichés dans le volet Sortie ne sont pas enregistrés sur le disque. À la place, MapForce crée des fichiers temporaires. Pour enregistrer la sortie, ouvrez la volet de Sortie et sélectionnez la commande de menu **Sortie | Enregistrer le fichier de Sortie** ou cliquez sur (**Enregistrer la sortie générée**) dans la barre d'outils.

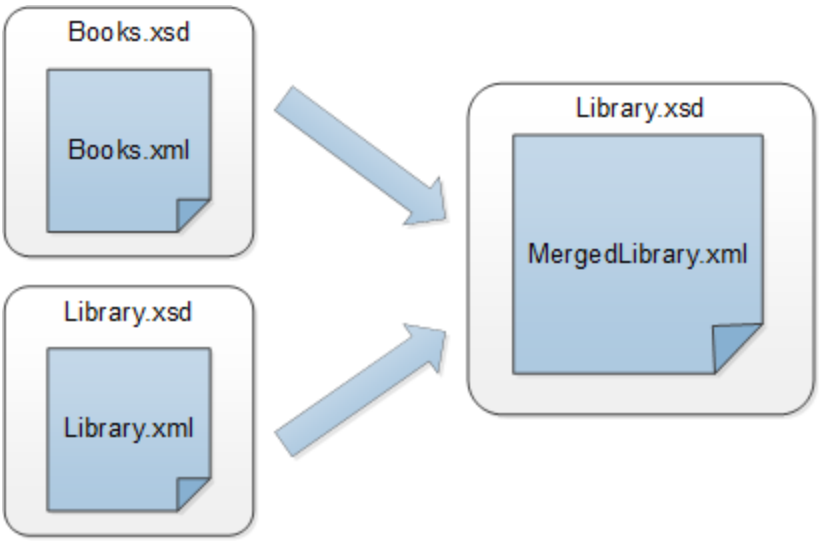
Pour configurer MapForce afin qu'il écrive la sortie directement sur les fichiers finaux au lieu des fichiers temporaires, rendez-vous sur **Outils | Options | Général**, et cochez la case *Écrire directement sur les fichiers de sortie finaux*. Veuillez noter qu'il n'est pas recommandé d'activer cette option pendant que vous suivez ce tutoriel, car vous risquez d'écraser les fichiers de tutoriel originaux par erreur.

consulter le code généré
Vous pouvez aussi consulter le code XSLT généré qui effectue la transformation. Pour conulter le code, ouvrez le volet XSLT3 situé en bas de la fenêtre de mappage. Afin de générer le code XSLT3 et de l'enregistrer dans un fichier, sélectionnez l'item de menu **Fichier | Générer code dans | XSLT 3.0**. Vous serez invité à choisir un dossier dans lequel le code généré sera enregistré. Une fois que la génération de code est achevée, le dossier de destination contiendra les deux fichiers suivants :

1. Un fichier de transformation XSLT, nommé après le schéma cible. Le fichier de transformation a le format suivant : **`MappingMapTo<TargetFileName>.xslt`**.
2. Un fichier **`DoTransform.bat`**, qui vous permet d'exécuter la transformation XSLT avec Altova RaptorXML Server depuis la ligne de commande. Pour exécuter la commande, vous allez devoir installer RaptorXML.

3.2 Sources multiples vers une cible

Dans ce tutoriel, vous apprendrez à fusionner les données depuis un nouveau fichier appelé **Library.xml** avec les données de **Books.xml**. Le résultat sera un fichier cible dénommé **MergedLibrary.xml**, qui contiendra les données des deux fichiers source. Le fichier cible sera basé sur le schéma **Library.xsd**. Veuillez noter que les fichiers source ont différents schémas. Si les fichiers source avaient le même schéma, vous pourriez aussi fusionner leurs données en utilisant une approche différente (voir Sources multiples à Cibles multiples 102). L'image ci-dessous représente un modèle abstrait de la transformation de données décrite dans le présent tutoriel.



La liste de codes ci-dessous montre l'extrait de **books.xml** qui sera utilisé en tant que première source de données.

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

La liste de code ci-dessous affiche un extrait de **Library.xml** qui sera utilisé comme seconde source de données :

```
<library>
   </publication>
      <id>5</id>
      <author>Alexandre Dumas</author>
      <title>The Three Musketeers</title>
      <Genre>Rock</Genre>
      <publish_year>1844</publish_year>
   </publication>
</ Library>
```

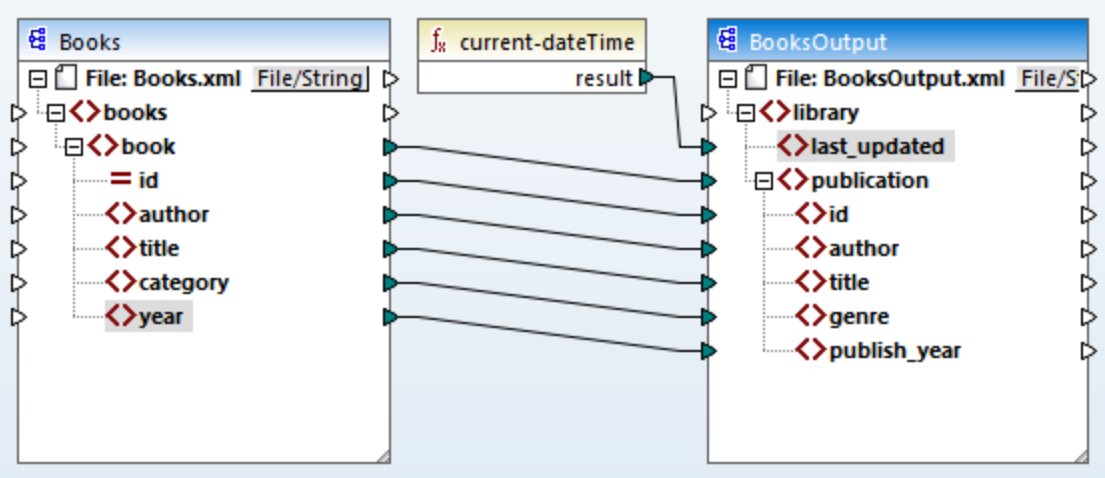
C'est ainsi que nous voulons que nos données fusionnées aient l'air dans le fichier cible dénommé **merged_library.xml** :

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

Pour effectuer la transformation, suivez les étapes décrites dans les sous-sections ci-dessous.

3.2.1 Préparer le design de mappage

Le point de départ de ce tutoriel est le mappage **Tut1_OneToOne.mfd** (*capture d'écran ci-dessous*) qui a été conçu dans le Tutoriel 1 [78]. Avant d'effectuer un changement au mappage, assurez-vous d'enregistrer ce design sous un nouveau nom dans le dossier **BasicTutorials**.

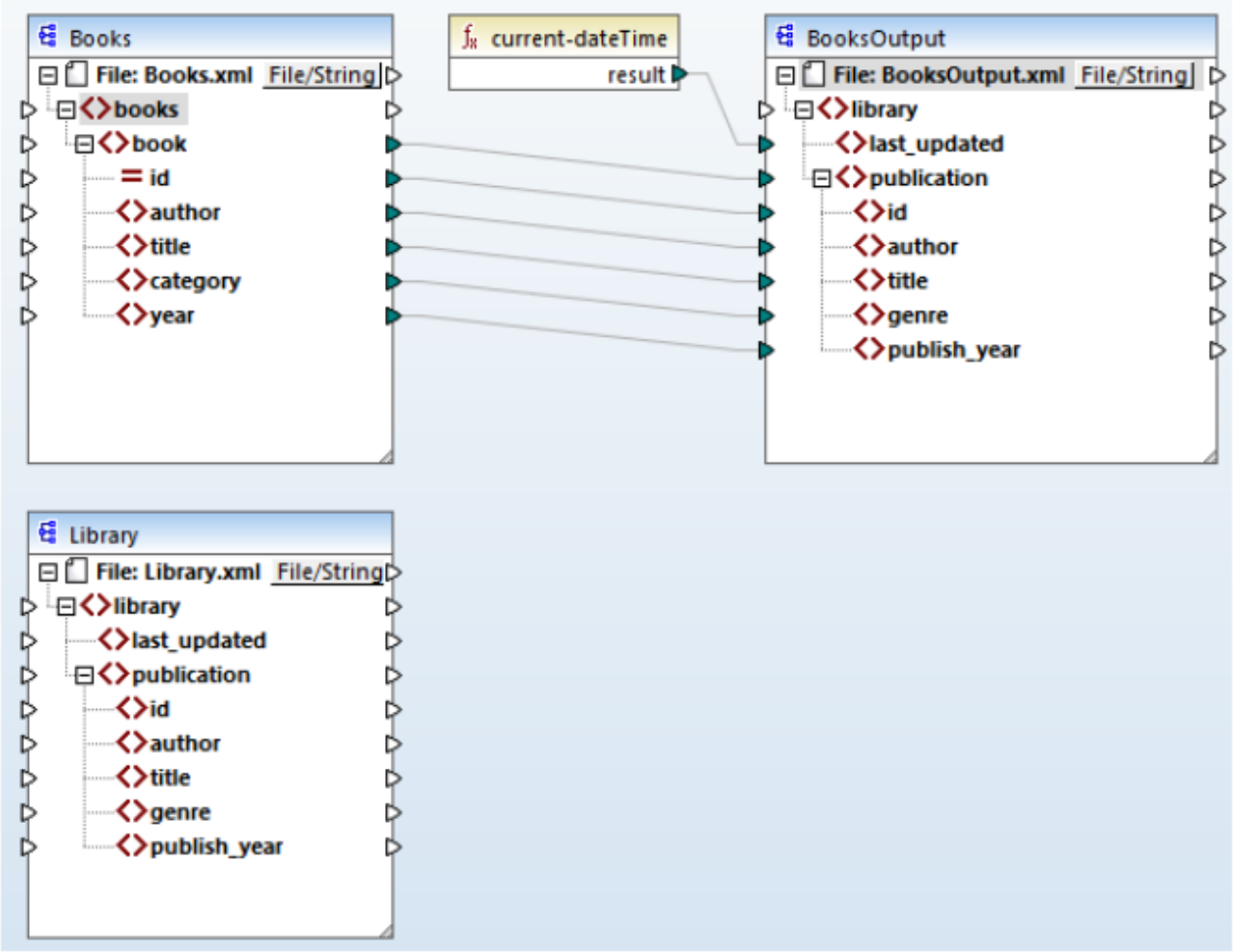


3.2.2 Ajouter seconde source

Dans la prochaine étape, nous allons ajouter un fichier de seconde source : Insérer **`Library.xml`** dans le mappage. Puisque le schéma (**`Library.xsd`**) est référencé dans ce fichier XML, vous ne devez pas ajouter le fichier de schéma dans une étape séparée. Pour vérifier si la référence de schéma est correcte, ouvrez les Paramètres de composant 113.

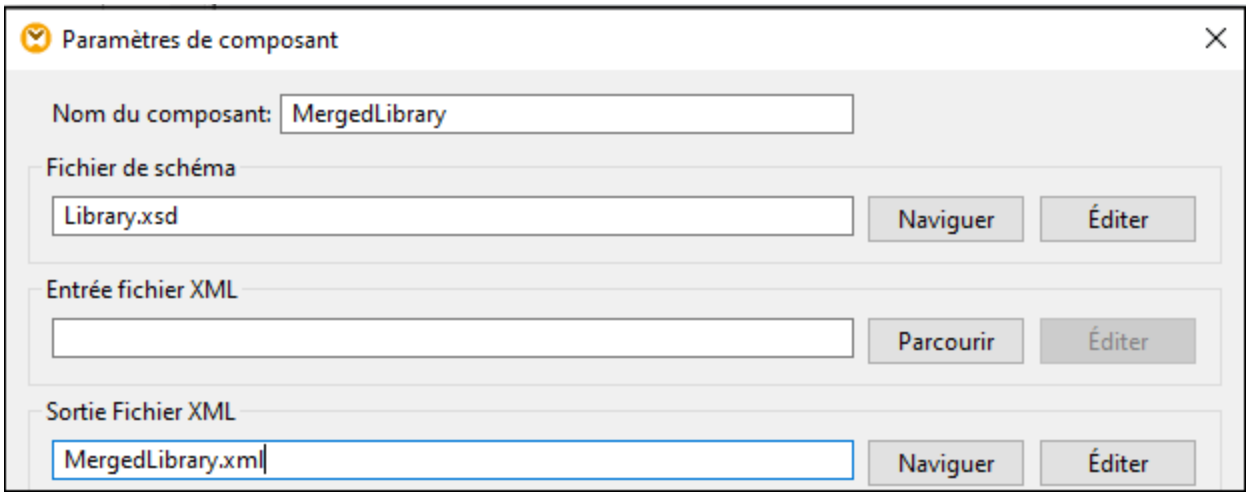
Puis cliquez et appuyez sur l'en-tête du nouveau composant et le glissez-le sous le composant `Books`. Vous pouvez toujours déplacer les composants dans toutes les directions. Néanmoins, le fait de placer un composant source à la gauche d'un composant cible vous permettra une lecture et une compréhension plus claires de votre mappage. Il s'agit là aussi de la convention utilisée pour tous les mappages illustrés dans cette documentation, ainsi que dans les fichiers de mappage échantillon qui accompagnent votre installation MapForce.

À cette étape, le design de mappage a l'air de ceci :



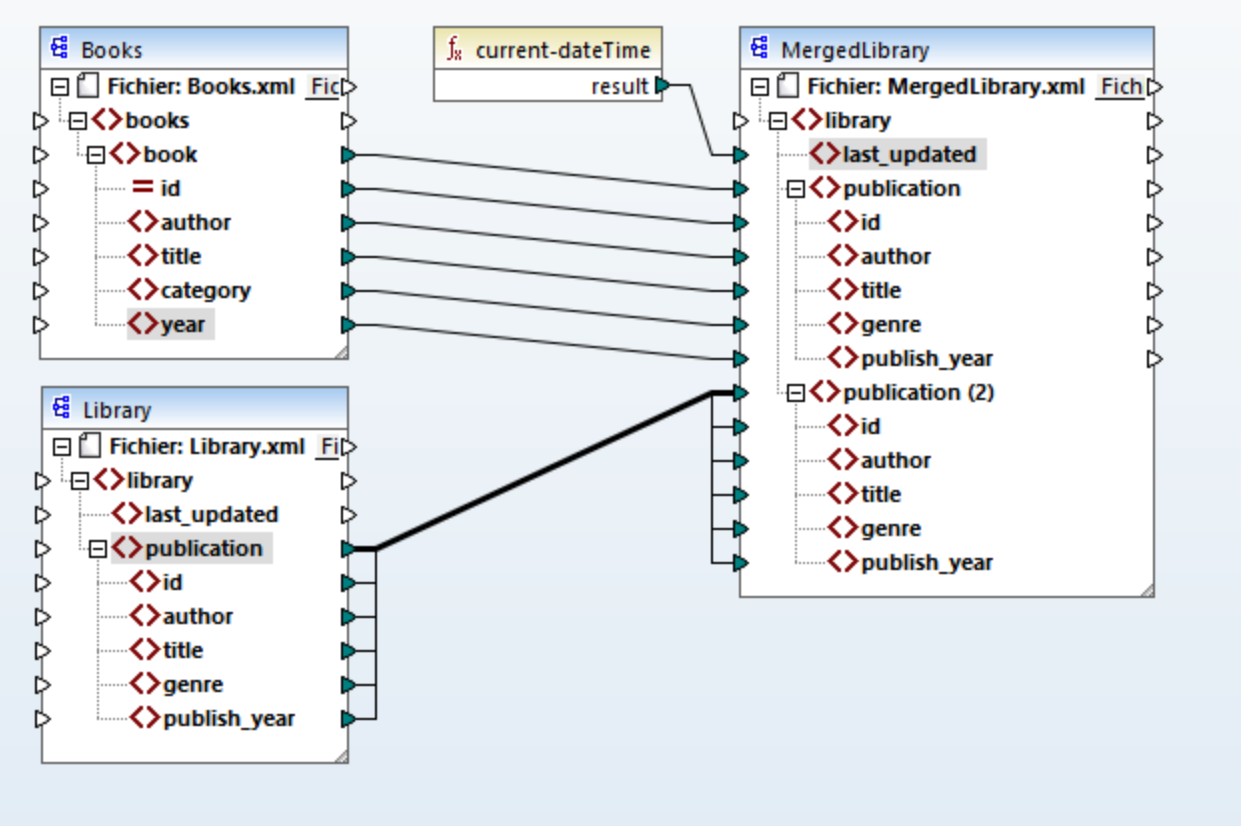
3.2.3 Configurer Sortie

À cette étape, le mappage est doté de deux composants source (`Books` et `Library`) et d'un composant cible (`BooksOutput`). À des fins de consistance et pour éviter toute confusion, nous devrons changer les paramètres du composant `BookOutput`. Double-cliquez sur l'en-tête du composant cible. Ceci ouvrira la boîte de dialogue Paramètres de composant[113]. Configurez les paramètres tels qu'indiqués ci-dessous.



3.2.4 Connecter Source et Cible seconde

La dernière étape du tutoriel est de connecter le deuxième composant source (`Library`) avec un composant cible (`MergedLibrary`). À cette fin, connectez l'élément `<publication>` dans **`Library.xml`** avec l'élément `<publication>` dans **`MergedLibrary.xml`**. Puisque le connecteur d'entrée cible a une connexion, MapForce vous invitera à remplacer cette connexion ou à dupliquer l'entrée. Dans ce tutoriel, notre objectif est de mapper des données depuis deux sources vers une cible. Cliquez donc sur **Doubler l'entrée**. Ce faisant,, vous configurez un composant cible de telle façon qu'il acceptera également les données également de la seconde source. À présent, le mappage ressemble à ceci :



La capture d'écran démontre que l'élément `publication` dans le composant cible a été doublé. Le nouveau nœud `publication(2)` acceptera les données provenant du composant source **`library`**. Veuillez noter que même si le nom de ce nœud apparaît en tant que `publication(2)` dans le mappage, son nom dans le fichier XML cible sera `publication`, qui est l'objectif que nous souhaitons atteindre.

Connexion copier-tout

Puisque les éléments enfant de l'élément `publication` dans le composant `Library` et l'élément `publication` dans le composant `MergedLibrary` ont les mêmes noms et types de données, ces éléments sont connectés par une ligne épaisse. Une telle connexion est appelée une *connexion copy-all* [55], qui rend le mappage plus compréhensible.

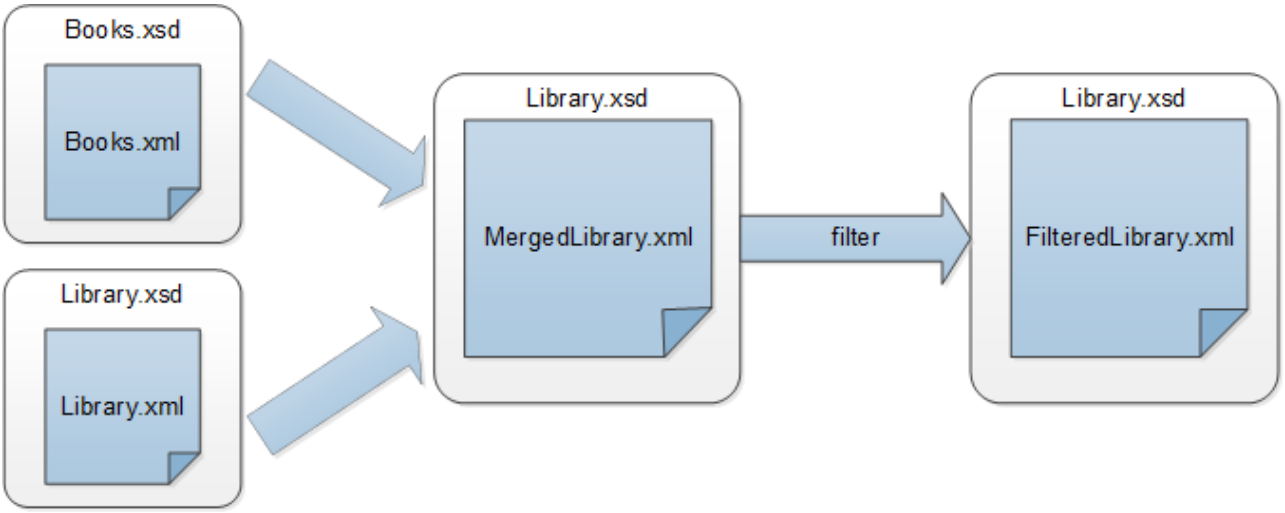
Consulter la sortie

Ouvrez le volet Sortie pour consulter le résultat. Vous remarquerez que les données provenant de **`Books.xml`** et **`Library.xml`** ont désormais été fusionnées dans le nouveau fichier **`MergedLibrary.xml`**. À votre convenance, le design de mappage est enregistré dans ce tutoriel en tant que **`Tut2_MultipleToOne.mfd`**. Ce mappage sera utilisé en tant que point de départ dans le prochain tutoriel [94].

3.3 Mappages en chaîne

Site web d'Altova : Mappage en chaîne Tutoriel vidéo

Ce tutoriel affiche comment fonctionne de multiples composants cible. L'objectif de ce tutoriel est de fusionner les données de deux fichiers source en un fichier cible, puis de filtrer les données de ce fichier cible et de transmettre ces données filtrées vers le second fichier cible. L'image ci-dessous illustre un modèle abstrait de la transformation de données décrite dans le présent tutoriel.



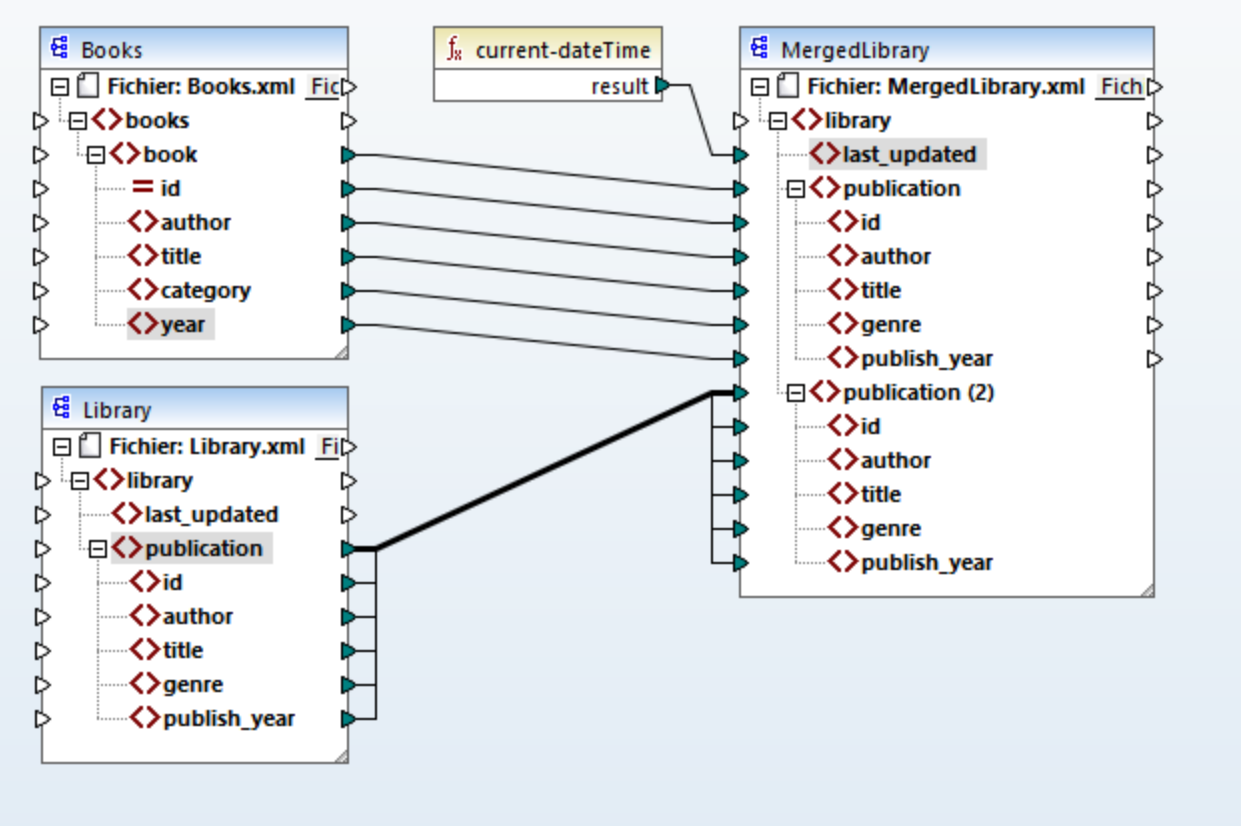
Dans le diagramme ci-dessus, les données sont tout d'abord fusionnées depuis deux fichiers source (**`Books.xml`** et **`Library.xml`**) dans un seul fichier cible appelé **`MergedLibrary.xml`**. Ensuite, les données sont transformées avec une fonction de filtre et sont passées au composant suivant appelé **`FilteredLlibrary.xml`**. Veuillez noter que **`FilteredLibrary.xml`** est basé sur le schéma **`Library.xsd`**. Le composant intermédiaire agit aussi bien en tant que cible et source des données. Dans MapForce, cette technique est appelée *mappage en chaîne*. Les mappages en chaîne permettent de consulter et enregistrer le/s résultat/s de mappage intermédiaire/s (dans notre cas, **`MergedLibrary.xml`**) et le résultat du dernier composant cible (dans notre cas, **`FilteredLibrary.xml`**).

Pour effectuer la transformation, suivez les étapes décrites dans les sous-sections ci-dessous.

Pour un autre exemple de mappage en chaîne, voir le tutoriel vidéo en haut de la page.

3.3.1 Préparer le design de mappage

Le point de départ de ce tutoriel **`Tut2_MultipleToOne.mfd`** (*voir la capture d'écran ci-dessous*). Ce mappage a été conçu dans le tutoriel précédent 89. Avant d'effectuer un changement au mappage, assurez-vous d'enregistrer ce design sous un nouveau nom dans le dossier **`BasicTutorials`**.

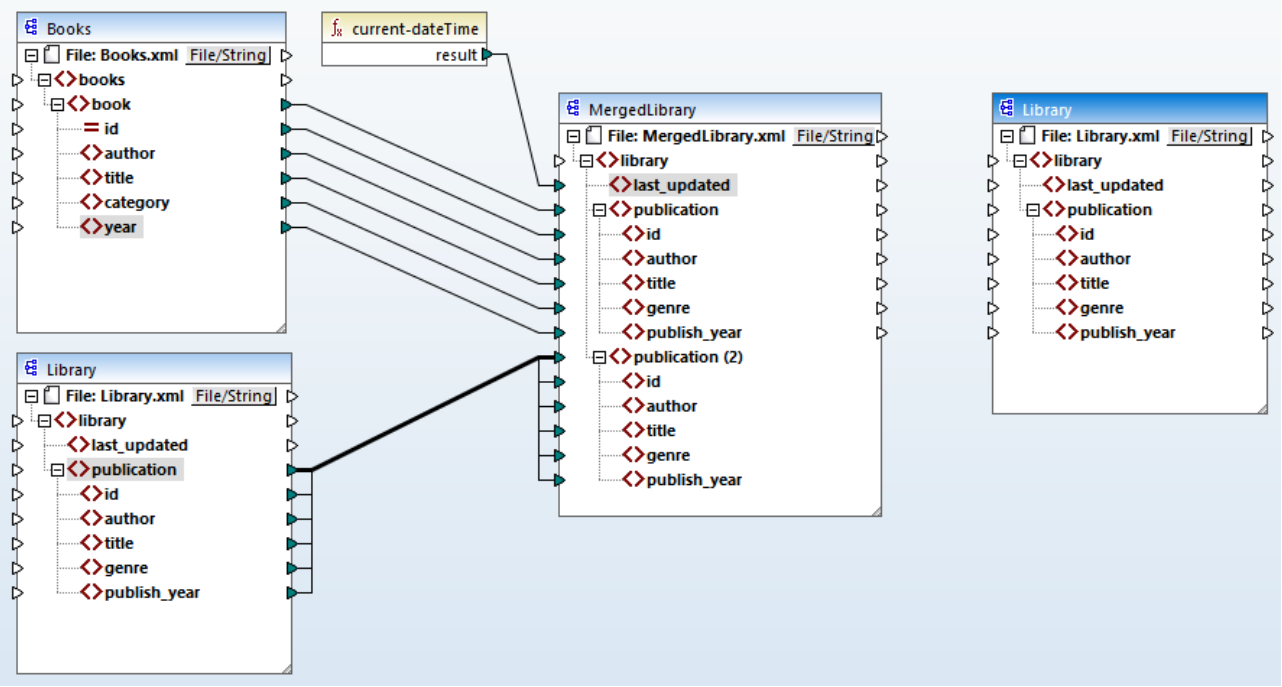


3.3.2 Configurer la deuxième cible

La prochaine étape est d'ajouter et de configurer le deuxième fichier cible, qui contiendra uniquement un sous-ensemble de données de la `publication` de **`MergedLibrary.xml`**.

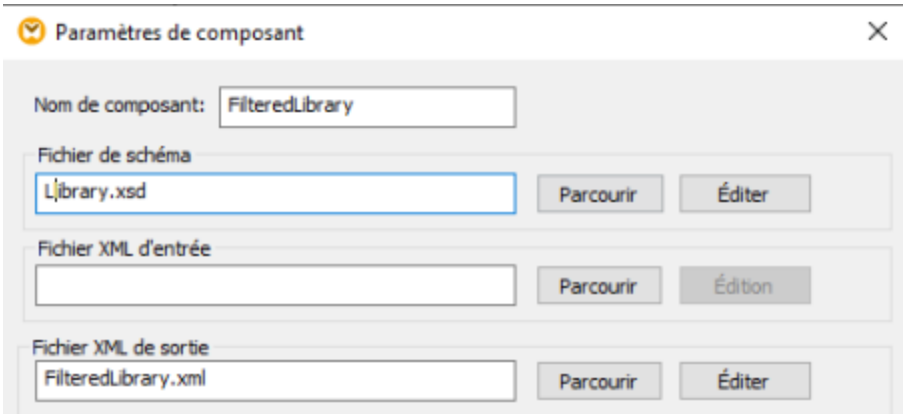
Ajouter le deuxième composant cible

Ajouter **`Library.xsd`** au mappage et cliquez **Skip** lorsque vous êtes invité à fournir un fichier d'instance échantillon. Notre deuxième composant cible n'a qu'une structure mais pas de contenu. À une étape ultérieure, nous allons mapper les données filtrées avec le fichier cible. Le design de mappage a désormais l'air de ceci :



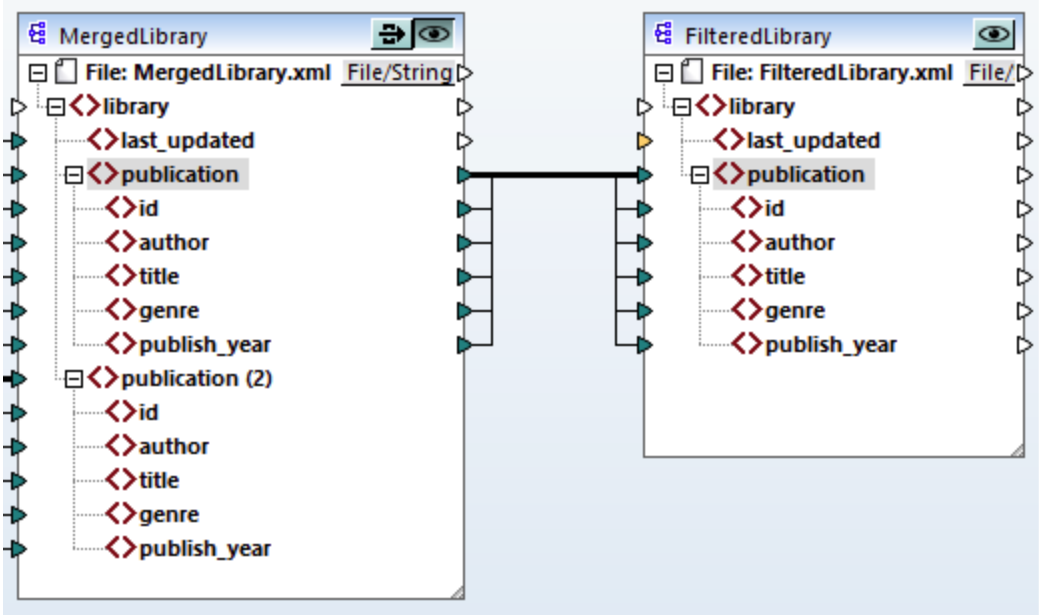
Configurer le deuxième composant cible

Comme affiché ci-dessus, le mappage a désormais deux composants source (`Books` et `Library`) et deux composants cible (`MergedLibrary` et `Library`). Pour éviter toute confusion, nous modifierons le nom du composant nouvellement ajouté `FilteredLibrary`. Pour ce faire, double-cliquez sur l'en-tête du composant le plus à droite et éditer les paramètres de composant[113] :



3.3.3 Connecter les Cibles

La prochaine étape consiste à mapper l'élément `publication` dans `MergedLibrary` dans l'élément `publication` dans `FilteredLibrary`. Quand vous connectez le connecteur de sortie de `MergedLibrary` avec un connecteur d'entrée de `FilteredLibrary`, MapForce vous informera que vous avez créé de multiples composants cible dans le mappage. Veuillez noter que des nouvelles touches sont maintenant disponibles dans le coin supérieur droit des deux composants cible : (**Preview**) et (**Pass-through**). Ces touches seront utilisées et expliquées dans les étapes suivantes.

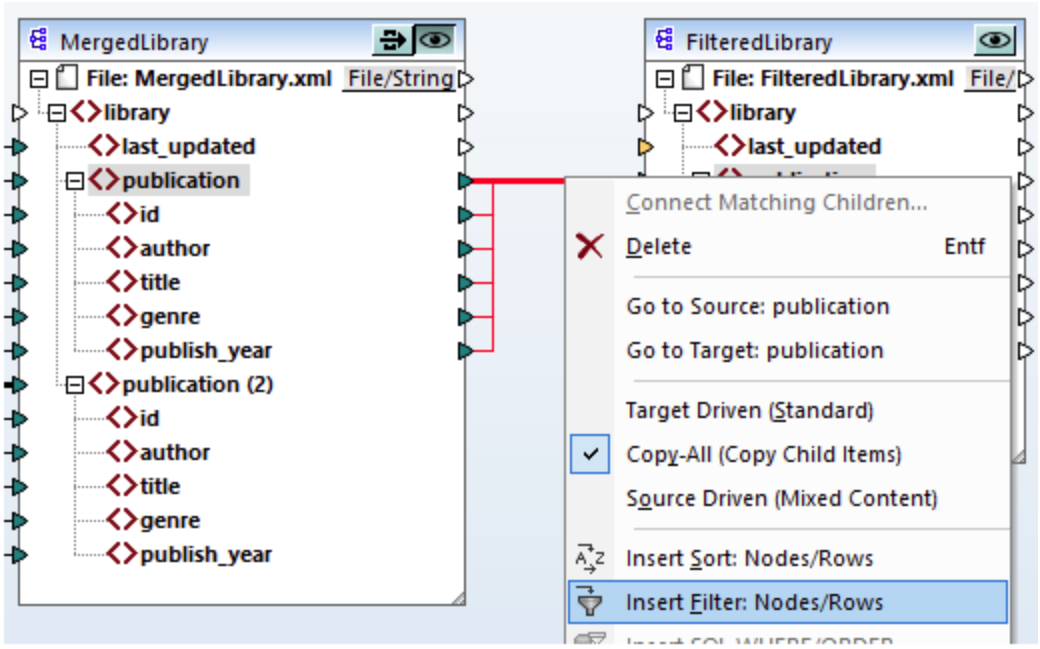


3.3.4 Filtrer les données

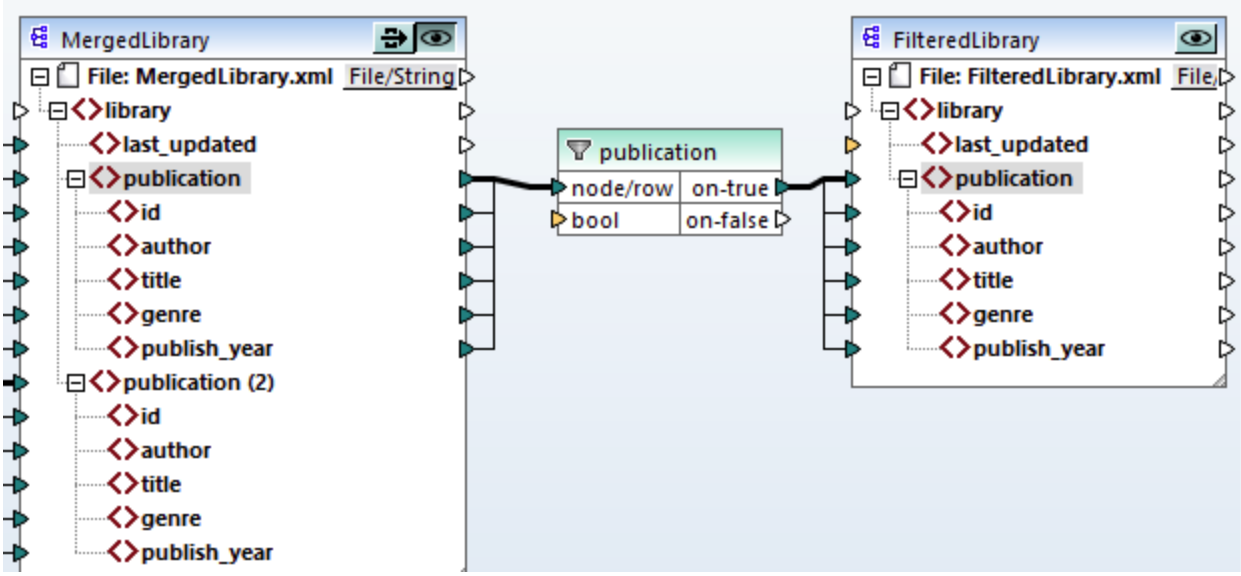
À cette étape, nous allons filtrer les données de `MergedLibrary` de telle façon que seuls les livres publiés après 1900 seront passés au composant `FilteredLibrary`. Nous allons utiliser le composant Filtre à cette fin.

Ajouter un filtre

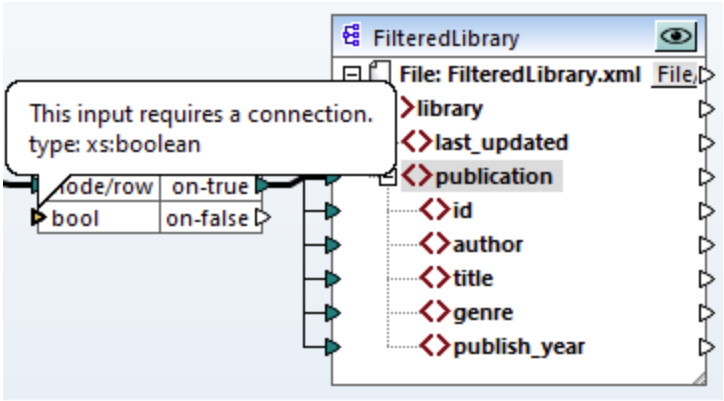
Pour ajouter un filtre, cliquez avec le bouton droit entre `MergedLibrary` et `FilteredLibrary` et sélectionnez **Insérer filtre : Nœuds/Lignes** depuis le menu contextuel(*capture d'écran ci-dessous*).



Le composant de filtre a désormais été ajouté au mappage (*voir la capture d'écran ci-dessous*).



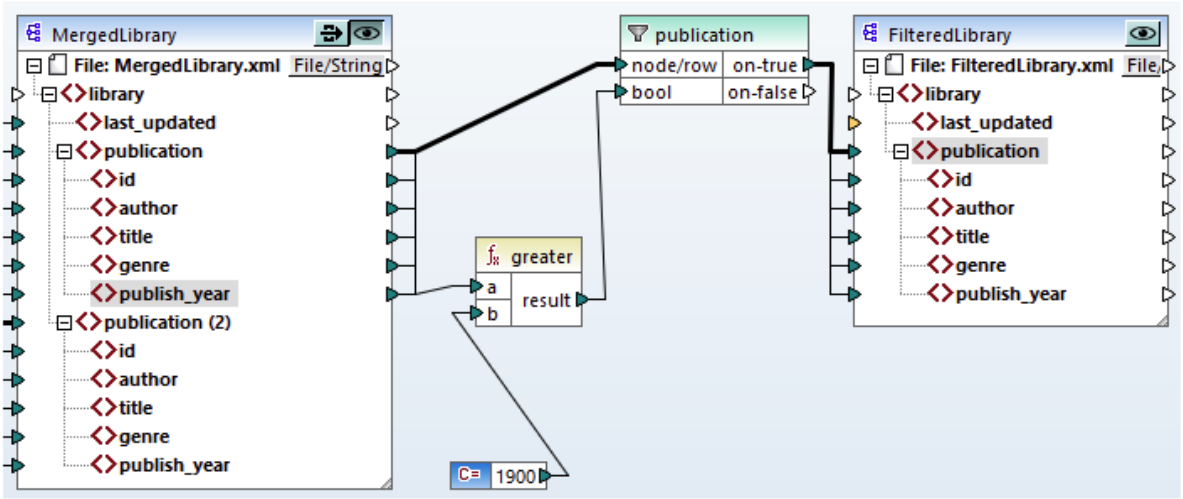
Dans la capture d'écran ci-dessus, le connecteur d'entrée `bool` est en surbrillance orange, ce qui signifie que cette entrée est obligatoire. Si vous passez sur un connecteur, vous pouvez voir si une entrée de type `xs:boolean` est requise (*voir la capture d'écran ci-dessous*). Pour voir les informations, cliquez sur (**Afficher infos**) dans la barre d'outils.



Que les livres après 1900

Le composant filtre requiert une condition qui renvoie `true` ou `false`. Quand la condition booléenne renvoie `true`, les données de la séquence actuelle `publication` seront copiées dans la cible. Lorsque la condition retourne `false`, les données ne seront pas copiées. Dans ce tutoriel, la condition requise est de mapper uniquement les livres qui ont été publiés après 1900. Pour créer la condition, suivez l'étape suivante :

1. Cliquez sur **Constant** dans la barre d'outils et tapez *1900* dans la barre de texte. Sélectionnez *Number* comme type.
2. Ajoutez la fonction **`greater`** au mappage.
3. Effectuez les connexions de mappage de et vers la fonction **`greater`**, tel qu'indiqué ci-dessous. La fonction **`greater`** comparera la valeur de l'élément `publish_year` de chaque publication avec la valeur de la constante. Seuls les enregistrements de publication dont l'année de publication est supérieure à 1900 seront mappés dans la cible.



3.3.5 Prévisualiser et enregistrer la sortie

Vous êtes désormais prêt à consulter et enregistrer la sortie des deux composants cible. Lorsqu'il y a de multiples composants cible dans le même mappage, chaque composant cible a la touche (**Aperçu**). Il n'est possible d'activer le Preview que d'un seul composant à la fois. En utilisant les boutons **Aperçu**, vous pouvez voir le résultat de mappage intermédiaire (dans notre exemple, les données appées dans le composant `MergedLibrary`) ainsi que le résultat final du mappage en chaîne (les données mappées dans le composant `FilteredLibrary`). Le composant `MergedLibrary` doit aussi avoir un bouton (**Pass-through**). La touche **Pass-through** contrôle comment la sortie sera générée (*voir les détails ci-dessous*).

Consulter et enregistrer les sorties intermédiaires et finales
Si vous voulez consulter et enregistrer la sortie des composants `MergedLibrary` et `FilteredLibrary`, suivez les étapes suivantes :

1. Cliquer sur la touche **Pass-through** dans le composant `MergedLibrary`.
2. Assurez-vous que la touche **Preview** du composant `FilteredLibrary` est aussi appuyée.
3. Ouvrez le volet Sortie. Les touches **Retour** et **Suivant** vous permettent de basculer entre les sorties.
4. Basculer la sortie que vous souhaitez enregistrer vers un fichier et cliquez sur **Enregistrer Fichier de sortie** dans la barre d'outils. Si vous souhaitez enregistrer les deux sorties, cliquez sur la touche **Enregistrer toutes les sorties générées** dans la barre d'outils.

Lorsque la fonction pass-through est active, le champ *Fichier XML d'entrée* du composant intermédiaire est désactivé automatiquement. Ceci garantit l'utilisation de la sortie générée quand vous consultez la portion du mappage entre les sources (`Books` et `Library`) et le première cible (`MergedLibrary`) par défaut comme entrée quand vous consultez la deuxième partie du mappage entre la première cible (`MergedLibrary`) et la deuxième cible (`FilteredLibrary`).

Consulter et enregistrer la sortie intermédiaire
Les composants intermédiaires dont la touche pass-through est activée ne peuvent pas être consultés. La section touche d'aperçu du composant intermédiaire est automatiquement désactivée car il n'est pas approprié de consulter les données et de les laisser passer en même temps. Si vous voulez consulter et enregistrer uniquement la sortie des composants (`MergedLibrary`), suivez les étapes suivantes :

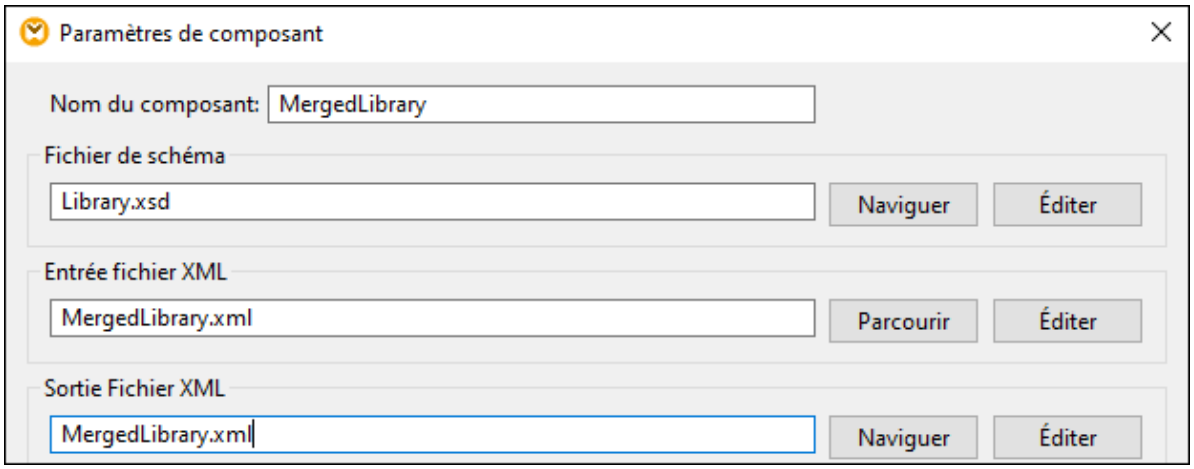
1. Désactivez la touche **Pass-though** du composant `MergedLibrary` si la touche a précédemment été activée.
2. Cliquez sur la touche **Aperçu** dans le composant `MergedLibrary` .
3. Ouvrez le volet Sortie.
4. Cliquez sur le bouton **Enregister fichier de sortie** dans la barre d'outils pour enregister la sortie vers un fichier.

Consulter et enregistrer la sortie finale
Si vous voulez consulter et enregistrer uniquement les données mappées depuis le composant intermédiaire vers le deuxième composant cible, suivez les instructions ci-dessous.

1. Désactivez la touche **Pass-though** du composant `MergedLibrary` si la touche a été précédemment activée.
2. Double-cliquez sur l'en-tête du composant `MergedLibrary`.
3. Assurez-vous de fournir le même nom du fichier dans le champ *Input XML File* et le *Output XML File* (*capture d'écran ci-dessous*). Lorsque vous désactivez la touche **Pass-through**, vous pouvez choisir quel fichier d'entrée devrait être lu par le composant intermédiaire. Dans la plupart des cas, ceci devrait être le même fichier que celui défini dans le champ *Output XML File*. Il est normalement sensé pour le

composant intermédiaire de recevoir un fichier à traiter et envoyer le même fichier au mappage suivant plutôt que de rechercher un nom de fichier différent.

Avoir le même fichier d'entrée et de sortie pour le composant intermédiaire est important quand la touche **Pass-through** pour le composant intermédiaire est désactivée. Ceci garantit l'utilisation de la sortie générée quand vous consultez la portion du mappage entre les sources (`Books` et `Library`) et le première cible (`MergedLibrary`) comme entrée quand vous consultez la deuxième partie du mappage entre la première cible (`MergedLibrary`) et la deuxième cible (`FilteredLibrary`). Si vous exécutez votre mappage avec MapForce Server (*Professional and Enterprise editions*) ou par le biais du code généré, les mêmes noms des fichiers d'entrée et de sortie du composant intermédiaire assurent que la chaîne de mappage n'est pas rompue. Sachez que, si les noms du fichier d'entrée et de sortie ne sont *pas* les mêmes au niveau du composant intermédiaire (si la touche **Pass-through** est inactive) peut entraîner des erreurs dans MapForce, le code généré, et dans l'exécution MapForce Server..



4. Cliquez sur la touche **Aperçu** du composant `FilteredLibrary`.
5. Ouvrez le volet Sortie.
6. Cliquez sur le bouton **Enregistrer fichier de sortie** dans la barre d'outils pour enregistrer la sortie vers un fichier.

Important

- La fonction pass-through est est disponible uniquement pour les composants basés sur fichier (par exemple, XML, CSV et texte. Les composants de base de données (*éditions Professional et Enterprise*) peuvent être intermédiaires, mais la touche pass-through n'est pas indiquée.
- Lorsque le mappage est exécuté par MapForce, le paramètre *Écrire directement sur le fichier de sortie final* (configuré depuis **Outils | Options | Généralités**[469]) détermine si les fichiers intermédiaires sont enregistrés en tant que fichiers temporaires ou en tant que fichiers physiques. Veuillez noter que cela est uniquement valide si le mappage est prévisualisé directement dans MapForce. Si ce mappage est exécuté par MapForce Server ou par le code généré, les fichiers actuels seraient produits à chaque étape dans le mappage en chaîne.

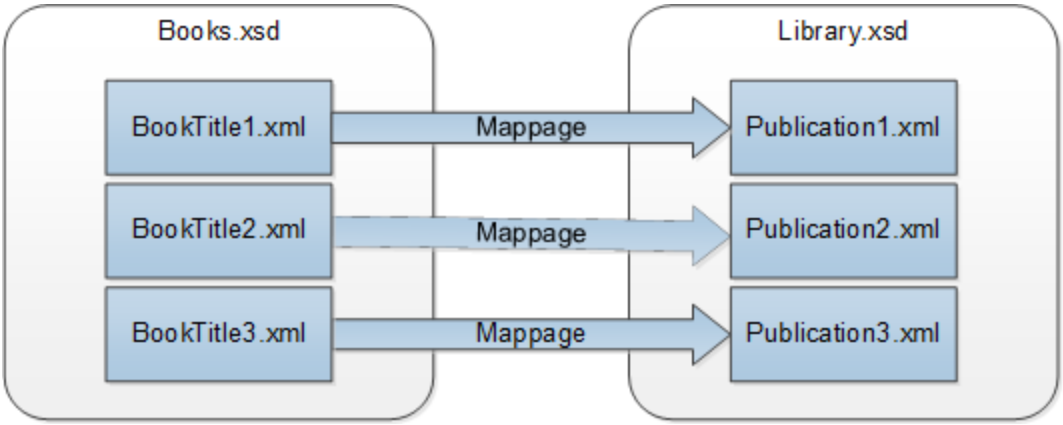
Vous avez désormais terminé de concevoir le mappage qui produit deux fichier de sortie. À votre convenance, le design de mappage est enregistré dans ce tutoriel en tant que **`Tut3_ChainedMapping.mfd`**.

3.4 Sources multiples vers cibles multiples

Ce tutoriel vous montre comment mapper les données depuis des fichiers source multiples vers des fichiers cible multiples. Pour illustrer cette technique, nous allons créer un mappage avec les objectifs suivants :

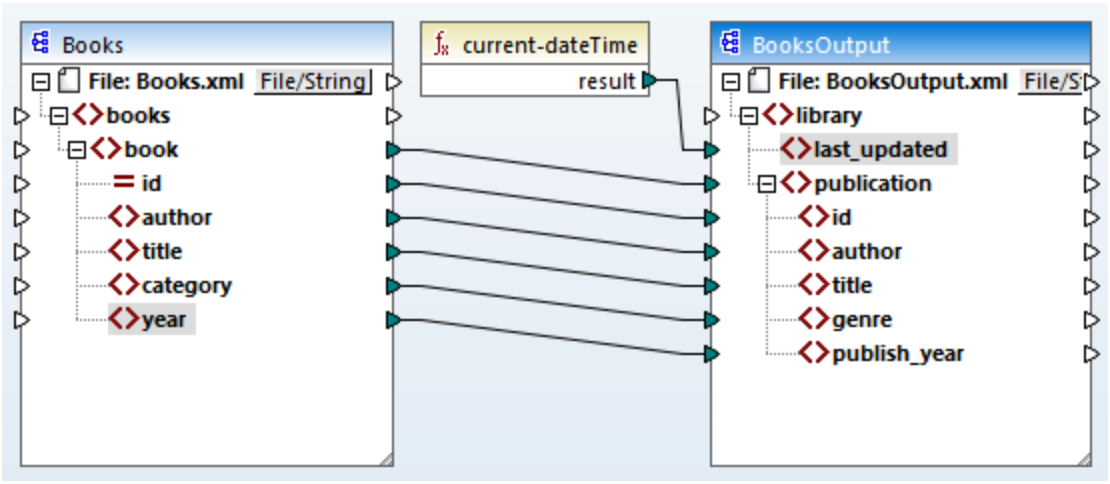
1. Pour lire des données depuis des fichiers XML multiples situé dans le même répertoire. Les fichiers sont basés dans le même schéma source.
2. Pour chaque fichier source XML, générez un nouveau fichier cible XML. Les fichiers cible seront basés dans un schéma cible.

L'image ci-dessous illustre un modèle abstrait de la transformation de données utilisée dans le présent tutoriel :



Présentation sommaire

Le point de départ de ce tutoriel est le mappage **`Tut1_OneToOne.mfd`** du premier tutoriel[78] (*voir la capture d'écran ci-dessous*).



Modifier le composant source
Nous modifierons les paramètres de composant du composant source pour qu'il lise les données depuis de multiples fichiers source : **BookTitle1.xml**, **BookTitle2.xml** et **BookTitle3.xml**. Chacun des trois fichiers est basé sur **Books.xsd** et entrepose un livre (*voir ci-dessous*).

BookTitle1.xml

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

BookTitle2.xml

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

BookTitle3.xml

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

Modifier le composant cible
Nous configurerons également le composant cible de telle manière que les données sont écrites dans de multiples fichiers cible. Les fichiers cible seront basés sur le même schéma appelé **Library.xsd**. Les fichiers cible générés seront appelés **Publication1.xml**, **Publication2.xml**, et **Publication3.xml** (*liste de code ci-dessous*).

Publication1.xml

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

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

Publication2.xml

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

Publication3.xml

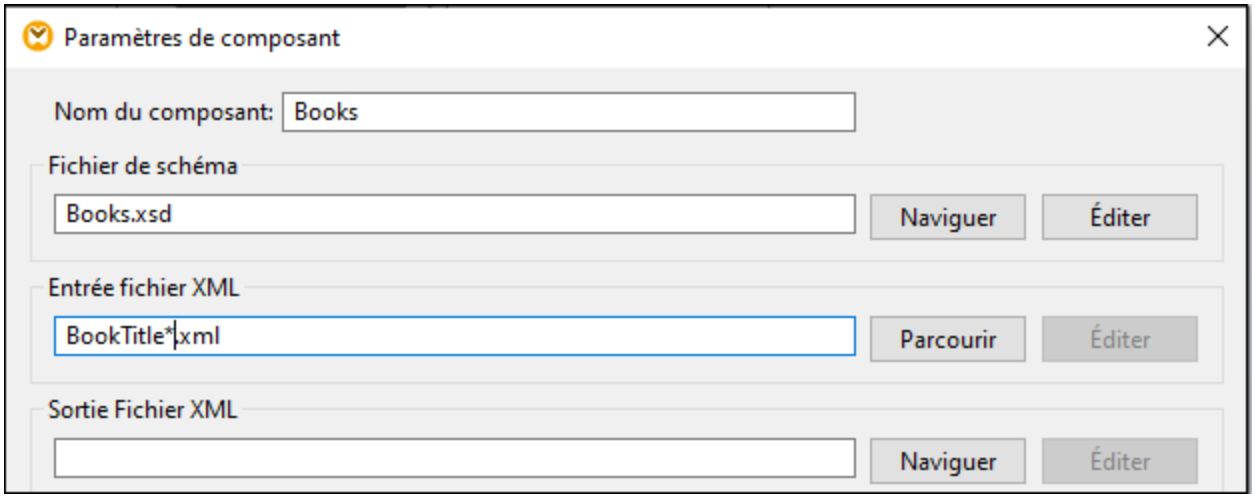
```
      <library>
         <publication>
            <id>3</id>
            <author>Herman Melville</author>
            <title>Moby Dick</title>
            <genre>Fiction</genre>
            <publish_year>1851</publish_year>
         </publication>
      </library>
```

Pour effectuer la transformation de données requise, suivez les étapes décrites dans les sous-sections ci-dessous.

3.4.1 Configurer l'entrée

La première étape est de modifier les paramètres de composant du composant source. Avant de changer les paramètres du composant, assurez-vous d'enregistrer votre mappage avec un nouveau nom dans le dossier **BasicTutorials**.

Pour donner l'instruction à MapForce pour traiter les multiples fichiers d'instance XML, double-cliquez sur l'en-tête et le type du composant **BookTitle*.xml** dans le champ *Entrée Fichier XML* (*capture d'écran ci-dessous*). L'astérisque (*) contenu dans le nom du fichier donne l'instruction à MapForce d'utiliser tous les fichiers avec le préfixe `BookTitle` en tant qu'entrées de mappage. Étant donné que le chemin est relatif, MapForce recherchera tous les fichiers `BookTitle` dans le même répertoire que le fichier de mappage. Vous pouvez aussi saisir une chemin absolu, le cas échéant.

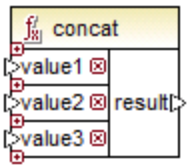


3.4.2 Configurer la Sortie, Partie 1

Dans cette étape, nous allons créer le nom de fichier de chaque fichier de sortie. Pour ce faire, nous utiliserons la fonction **concat** 307 qui relie toutes les valeurs qui lui sont fournies. Lorsque ces valeurs sont réunies, elles créeront un nom de fichier de sortie (par ex., **Publication1.xml**). Pour générer les nom de fichier en utilisant la fonction **concat**, suivez les étapes suivantes :

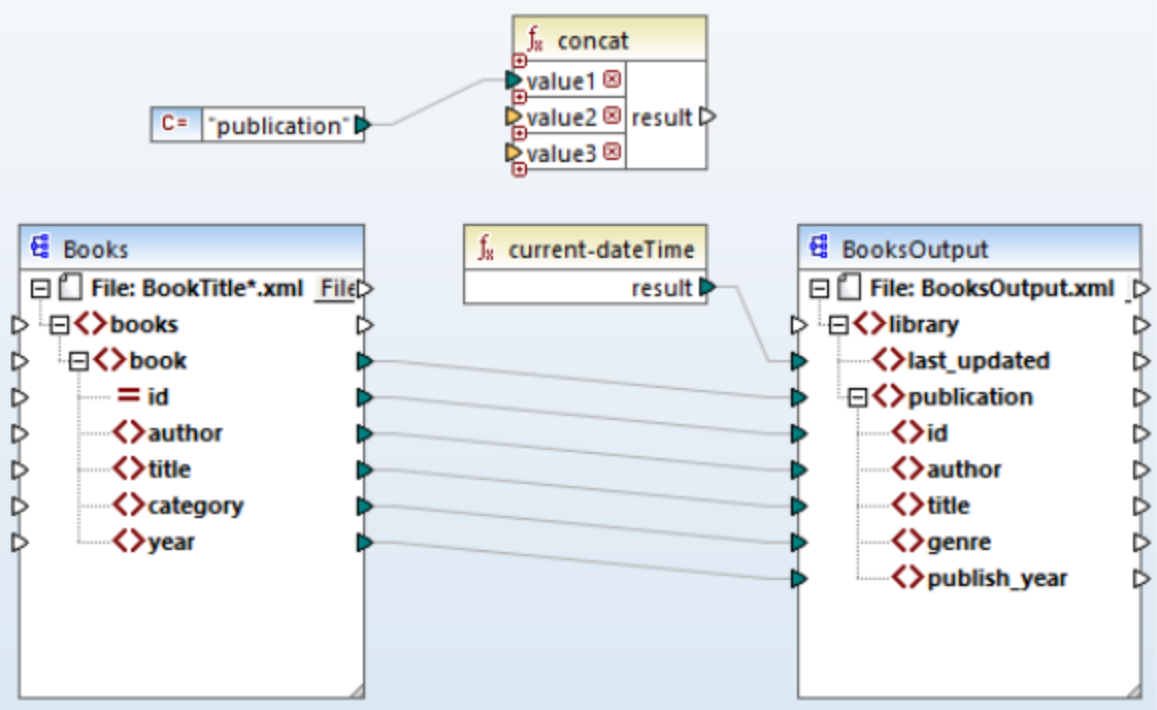
Étape 1 : Ajouter la fonction de concaténation

Ajoutez 83 la fonction **concat** (*capture d'écran ci-dessous*) à la zone de mappage. Par défaut, cette fonction a deux paramètres quand elle est ajoutée au mappage. Dans notre exemple, nous avons besoin de trois paramètres. Cliquez sur ⊡ (**Ajouter paramètre**) à l'intérieur du composant de la fonction et ajoutez-y un troisième paramètre. Veuillez noter qu'en cliquant sur ⊠ (**Supprimer paramètre**), ceci supprime un paramètre.



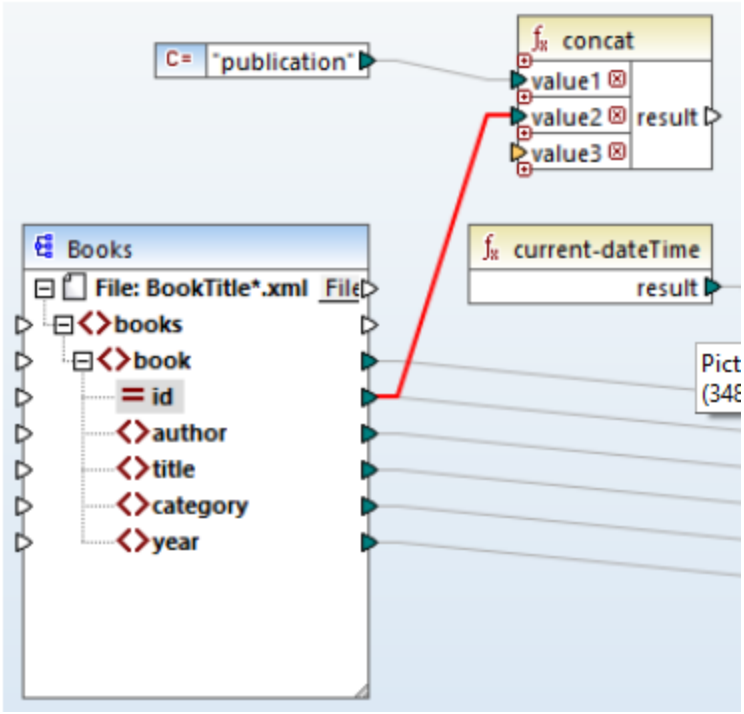
Étape 2 : Insérer une constante

La prochaine étape est d'ajouter une constante. Lorsque vous êtes invité à fournir une valeur, saisissez `publication` et laissez l'option *String* inchangée. Connectez la constante avec la `value1` de la fonction **concat**, telle qu'affichée dans la capture d'écran ci-dessous :



Étape 3 : Fournir des ID

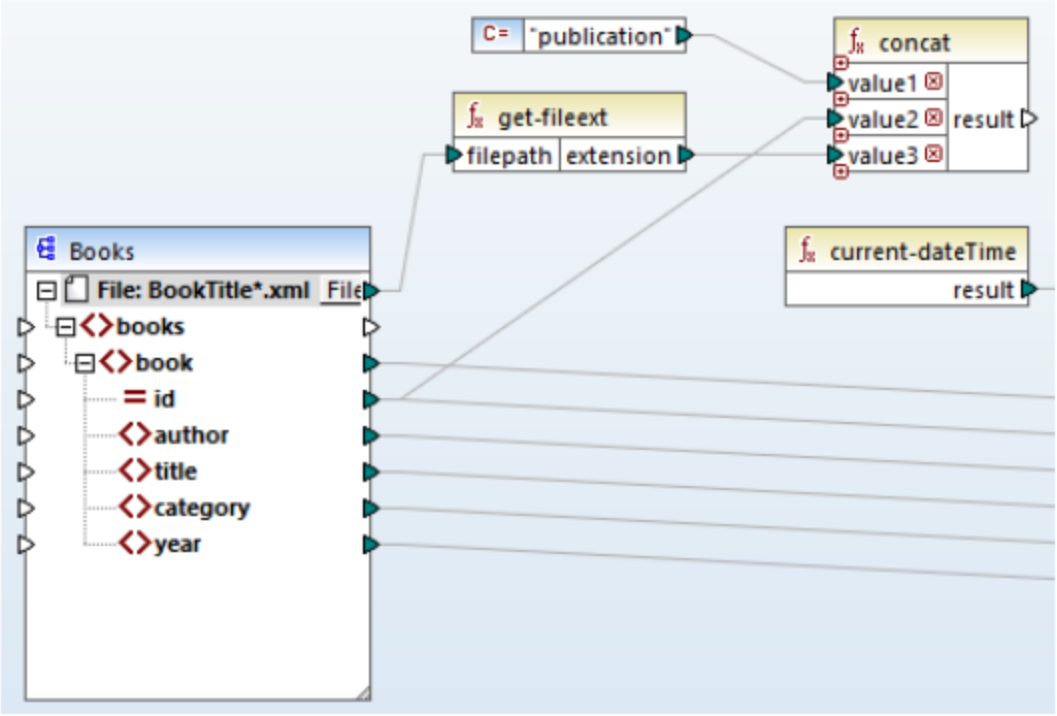
Connectez l'attribut `id` du composant source avec `value2` de la fonction **`concat`** (*capture d'écran ci-dessous*) L'attribut `id` du fichier XML source fournit une valeur d'identifiant unique pour chaque fichier. Cela sert à empêcher que les fichiers soient générés avec le même nom. La connexion devient rouge lorsque vous cliquez sur celle-ci.



Étape 4 : Extraire l'extension de fichier

Ajouter **`get-fileext`** [252] dans la zone de mappage. Créez une connexion depuis le nœud supérieur du composant source (**`Fichier : BookTitle*.xml`**) vers le paramètre `filepath` de cette fonction (*capture d'écran ci-dessous*).

La prochaine étape est de se connecter au paramètre `extension` de la fonction **`get-fileext`** vers la fonction `value3` de la fonction **`concat`**. Ce faisant, vous extrayez uniquement la partie de l'extension (dans ce cas, **`.xml`**) depuis le nom de fichier source et vous la passez au nom de fichier de sortie.

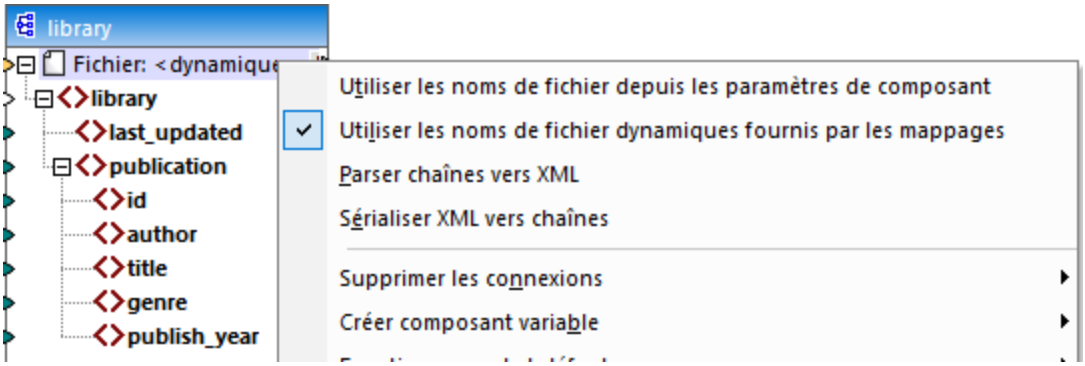


3.4.3 Configurer la Sortie, Partie 2

Dans la deuxième partie de la configuration de sortie, nous donnons l'instruction à MapForce de générer des fichiers de sortie de manière dynamique, ce qui signifie que chaque fichier de sortie recevra son nom basé sur les arguments fournis à la fonction **concat**. Pour ce faire, nous utiliserons des noms de fichier dynamiques (*voir les sous-sections ci-dessous*). Pour plus d'informations concernant les noms de fichier dynamiques, voir Traiter plusieurs fichiers d'entrée et de sortie 400.

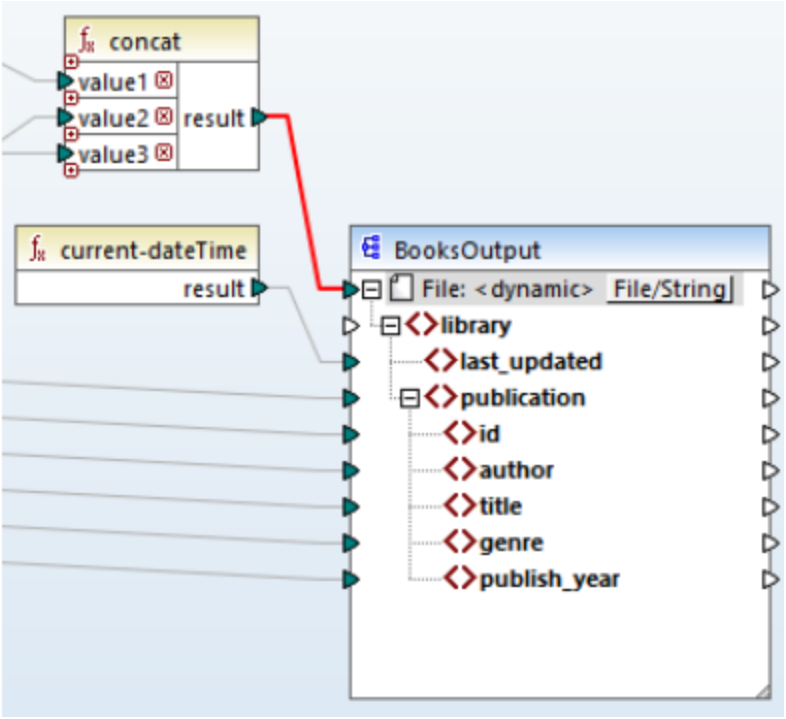
Étape 1 : Définir noms de fichiers dynamiques

Pour donner l'instruction à MapForce de générer les fichiers d'instance de manière dynamique, cliquez sur File ou File/String à côté du nœud `Fichier` du composant cible et sélectionnez **Utiliser Noms de fichiers dynamiques fournis par le mappage** depuis le menu contextuel (*capture d'écran ci-dessous*).



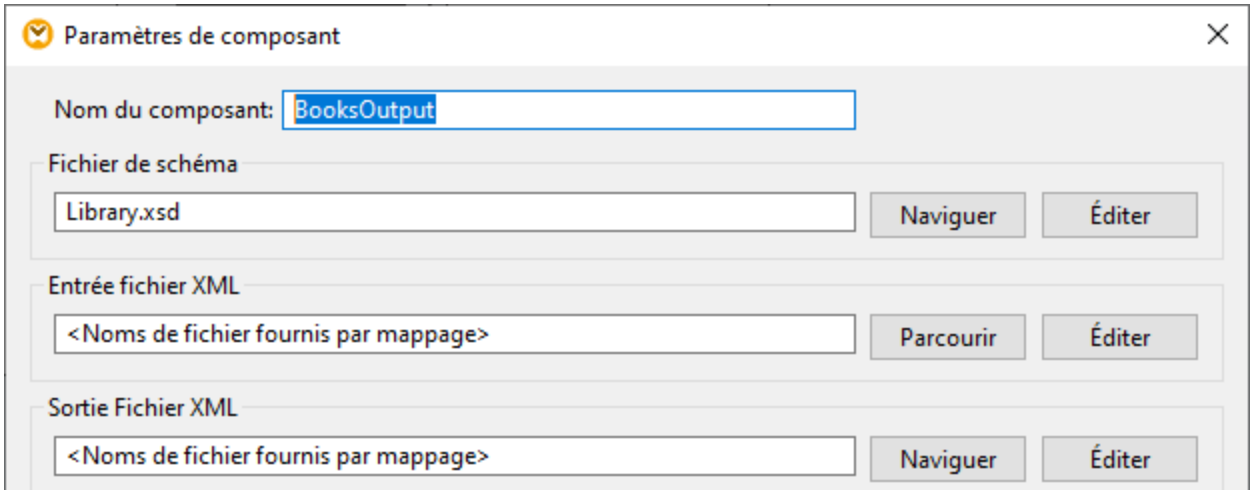
Étape 2 : Connecter fonction concat et nœud dynamique

Nous allons donc connecter le résultat de la fonction **`concat`** avec le `Fichier : Nœud <dynamic>` du composant cible (*capture d'écran ci-dessous*).



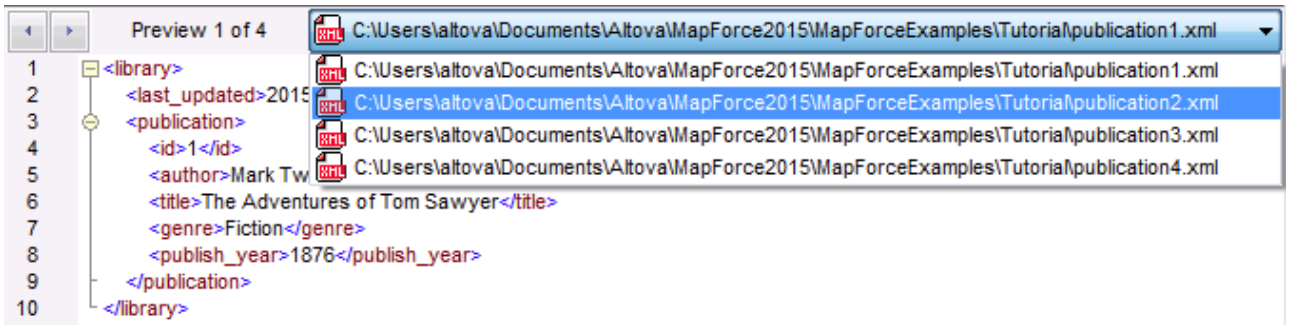
Étape 3 : Vérifier paramètres de composant

Dans les paramètres du Composant, vous noterez que les zones de texte *Entrée Fichier XML* et *Sortie Fichier XML* sont désactivées et affichent <*File names supplied by the mapping*> (*capture d'écran ci-dessous*). Ceci est une indication que vous avez fourni les noms de fichier d'instance de manière dynamique du mappage. Pour cette raison, il n'est plus possible de les définir dans les paramètres de composant.



Étape 4 : Générer des fichiers de sortie

Vous pouvez à présent exécuter le mappage et voir le résultat, ainsi que le nom des fichiers générés. Vous pouvez naviguer à travers les fichiers de sortie en utilisant les touches gauche et droite dans le coin supérieur gauche du volet Sortie ou en cliquant sur un fichier depuis la liste déroulante (*la capture d'écran ci-dessous*).



À votre convenance, le design de mappage est enregistré dans ce tutoriel en tant que **`Tut4_MultipleToMultiple.mfd`**.

4 Composants de structure

Cette section fournit des informations sur différents formats de données que vous pouvez utiliser comme sources et cibles de données :

- XML et Schéma XML 112

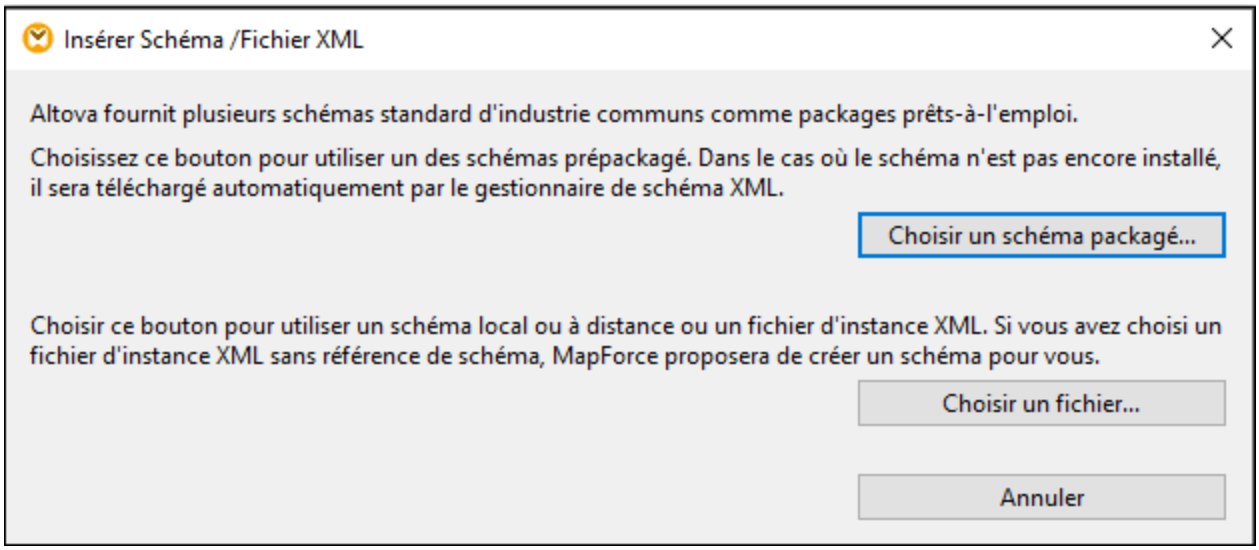
4.1 XML et schéma XML

Site web d'Altova : Mappage XML

XML est un langage de balisage pour les documents de texte. XML Schema définit la structure et les contraintes de documents XML Dans MapForce, les fichiers XML sont des composants structurels[32] qui peuvent être utilisés comme sources de données et cibles. Pour des informations sur les scénarios de transformation des données de base, voir Tutoriels[77].

Insérer schéma/fichier XML

Pour insérer un schéma/fichier XML, sélectionnez la commande de menu **Insérer | Schéma/Fichier XML** ou le bouton de la barre d'outils  . La boîte de dialogue (*voir la capture d'écran ci-dessous*) vous invitera à choisir entre un schéma de norme industrielle packagé et un fichier de schéma/d'instance local ou à distance. Si vous choisissez un schéma packagé, vous serez invité à sélectionner un point d'entrée. Si le schéma que vous souhaitez utiliser n'est pas encore installé, vous serez redirigé vers le Gestionnaire de taxonomies XML[129] pour télécharger.



Générer un Schéma XML
Lorsque vous ajoutez un fichier XML local ou à distance sans référence de schéma, MapForce suggèrera générer un schéma XML pour vous. Vous serez ensuite invité à sélectionner le répertoire dans lequel le schéma généré devrait être enregistré.

Lorsque MapForce génère un schéma depuis un fichier XML, les types de données pour des éléments/attributs doivent être inférés depuis le document d'instance XML et ne sont pas exactement ce à quoi vous vous attendez. Nous vous recommandons de contrôler si le schéma généré est une représentation précise des données d'instance.

Si les éléments ou les attributs dans plus d'un espace de noms sont présents, MapForce génère un Schéma XML séparé pour chaque espace de noms distinct ; c'est pourquoi plusieurs fichiers peuvent être créés sur le disque.

DTD comme structure du document
À partir de MapForce 2006 SP2, les DTD namespace-aware sont pris en charge pour les composants source et cible. Pour rendre les mappages possibles, les URI d'espace de nom sont extraits depuis les déclarations d'attribut de DTD `xmlns`. Toutefois, certains DTD contiennent des déclarations d'attribut `xmlns*` sans URI d'espace de nom (par ex., des DTD utilisés par StyleVision). Pour rendre ces DTD utilisables dans MapForce,définissez l'attribut `xmlns` avec l'URI d'espace de noms comme indiqué ci-dessous :

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

Notes relatives aux valeurs d'énumération

Pour les nœuds dont les types de données ont des facettes d'énumération, vous pouvez créer une Value-Map qui aura toutes les valeurs d'énumération pré-remplies. Ceci rend le traitement et le mappage des valeurs d'énumération plus facile. Pour plus d'informations, voir Value-Maps 186.

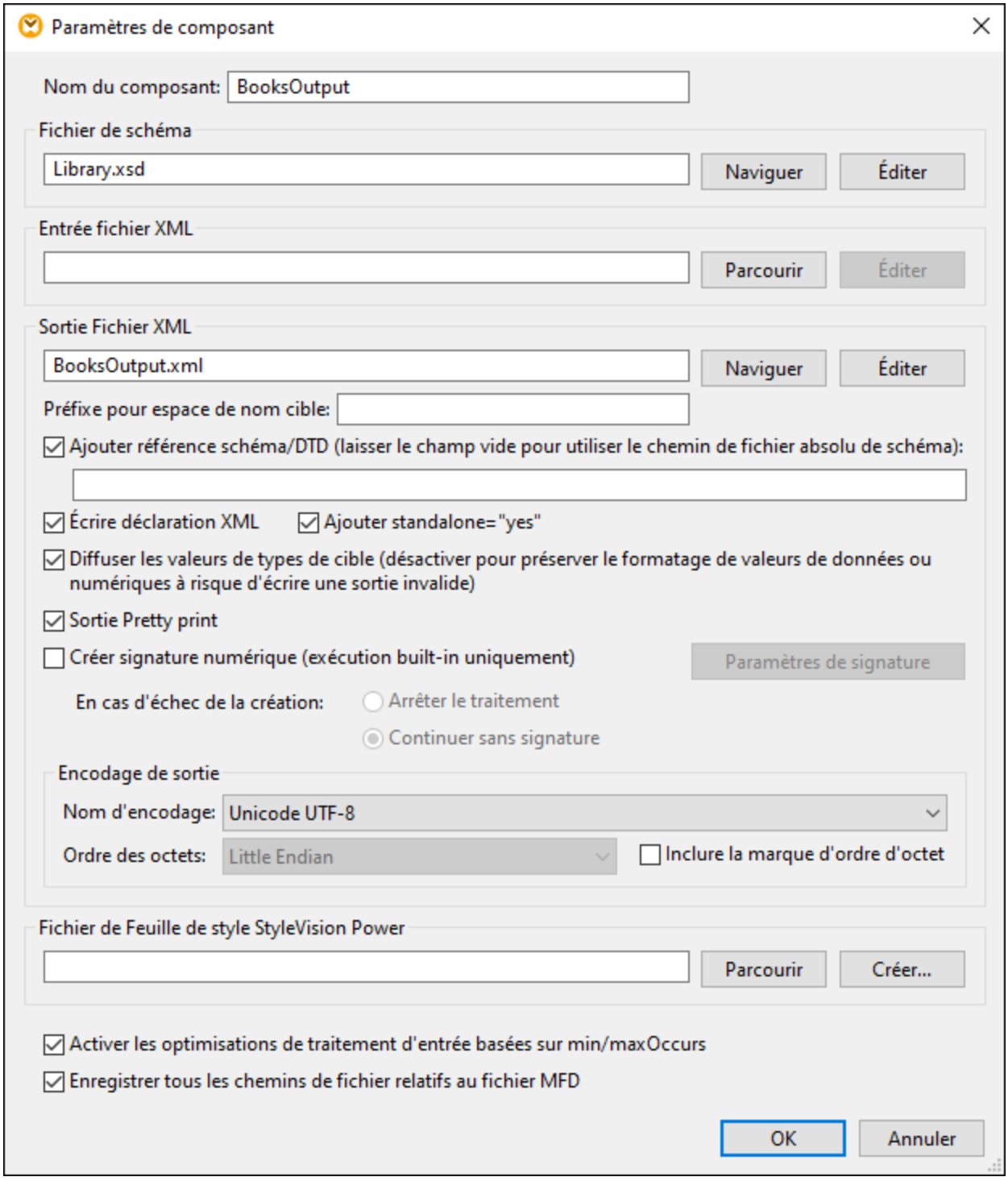
Dans cette section

La section est organisée en rubriques suivantes :

4.1.1 Paramètres de composant XML

Après avoir ajouté un composant XML dans la zone de mappage, vous pouvez configurer son paramètres applicables depuis le dialogue **Paramètres de composant** (*voir la capture d'écran ci-dessous*). Vous pouvez ouvrir le dialogue **Paramètres de composant** d'une des manières suivantes :

- En double-cliquant sur l'en-tête du composant
- En cliquant avec la touche de droite sur l'en-tête du composant et sélectionnez **Propriétés**
- En sélectionnant le composant dans le mappage et cliquez sur les **Propriétés** dans le menu **Composant**



Les paramètres disponibles sont décrits dans les sous-pages ci-dessous.

Paramètres généraux

- Nom de composant

Le nom de composant est généré automatiquement lorsque vous créez un composant. Néanmoins, vous pouvez changer le nom à tout moment. Le nom de composant peut contenir des espaces et les caractères de point final. Le nom de composant ne doit pas contenir barres obliques, des barres obliques inversées, des points-virgule, des guillemets doubles, des espaces de début et de fin. Si vous souhaitez changer le nom du composant, veuillez noter :

- Si vous souhaitez déployer le mappage vers FlowForce Server, le nom du composant doit être unique.
- Il est recommandé de n'utiliser que des caractères qui peuvent être saisis dans la ligne de commande. Les caractères de types nationaux peuvent présenter des encodages différents dans Windows et dans la ligne de commande.

Fichier de schéma

Spécifie le nom ou le chemin du fichier de schéma XML utilisé par MapForce pour valider et mapper des données. Pour changer de fichier de schéma, cliquez sur **Parcourir** et choisissez un nouveau fichier. Pour éditer le fichier dans Altova XMLSpy, cliquez sur **Éditer**.

Fichier d'entrée XML

Spécifie le fichier d'instance XML à partir duquel MapForce lira des données. Ce champ est important pour un composant de source et est rempli lorsque vous avez tout d'abord créé le composant et l'attribuez à un fichier d'instance XML. Dans le composant source, le nom de fichier d'instance est également utilisé pour détecter l'élément racine XML et le schéma référencé, et pour valider par rapport au schéma sélectionné. Pour changer de fichier de schéma, cliquez sur **Parcourir** et choisissez un nouveau fichier. Pour éditer le fichier dans Altova XMLSpy, cliquez sur **Éditer**.

Fichier de sortie XML

Spécifie le fichier d'instance XML à partir duquel MapForce écrira des données. Ce champ est significatif pour un composant cible. Pour changer de fichier de schéma, cliquez sur **Parcourir** et choisissez un nouveau fichier. Pour éditer le fichier dans Altova XMLSpy, cliquez sur **Éditer**.

Préfixe pour l'espace de noms cible

Vous permet de saisir un préfixe pour l'espace de nom cible. Avant d'assigner le préfixe, assurez-vous que l'espace de nom cible est défini dans le schéma cible.

Ajouter une référence schéma/DTD

Ajoute le chemin du fichier de schéma XML référencé vers l'élément racine de la sortie XML. Le chemin de schéma saisi dans ce champ est écrit dans un/des fichier(s) d'instance cible généré(s) dans l'attribut `xsi:schemaLocation` ou dans la déclaration `DOCTYPE` si un DTD est utilisé.

MapForce Professional et Enterprise Edition : Si vous générez du code dans XQuery ou C++, ajouter la référence DTD n'est pas pris en charge.

La saisie d'un chemin dans ce champ vous permet de définir où le fichier de schéma référencé par le fichier d'instance XML devra se trouver. Cela permet de garantir que l'instance de sortie peut être validée à la destination de mappage lorsque le mappage est exécuté. Vous pouvez saisir une adresse **`http://`** dans ce champ, ainsi qu'un chemin relatif ou absolu.

Désactiver cette option vous permet de déconnecter l'instance XML depuis le schéma XML référencé ou le DTD. Ceci peut être utile, par exemple, si vous voulez envoyer la sortie XML à quelqu'un qui n'a pas accès au schéma XML sous-jacent.

- Écrire la déclaration XML

Par défaut, l'option est activée, ce qui signifie que la déclaration XML est écrite dans la sortie. La table ci-dessous montre comment cette fonction est prise en charge dans les langages cibles de MapForce et les moteurs d'exécution.

Langage cible/ Moteur d'exécution	**Lorsque la sortie est un fichier**	**Lorsque la sortie est un string**
Built-In *éditions Professional et Enterprise*)	Oui	Oui
MapForce Server *Professional et Enterprise editions*)	Oui	Oui
XQuery (*éditions Professional et Enterprise*), XSLT	Oui	Non
Générateur de code (C++, C#, Java) (*Professional et Enterprise editions*)	Oui	Oui

- Add standalone="yes"

Sélectionner cette option insert la déclaration `standalone="yes"` dans la déclaration XML de votre fichier XML cible. Pour plus d'information, voir *Déclaration de Document autonome*.

Veuillez prendre note des points suivants :

- Quand l'option *standalone="yes"* est sélectionnée, la génération de sortie est compatible avec XSLT 1-3, Built-In, et le code généré (C#, Java, C++ MSXML, C+ Xerces). Le langage de transformation Built-In[17] et le code généré sont disponibles dans les éditions Professional et Enterprise.
- Il n'existe pas de prise en charge pour les champs de bases de données XML encastrés et requêtes de services web (*éditions Professional et Enterprise*).

- Convertir les valeurs en types de cible

Cette option vous permet de définir() les types de schéma XML cible doivent être utilisés du mappage, ou(ii) si les données mappées vers le composant cible doivent être traitées en tant que valeurs string. Par défaut, ce paramètre est activé. La désactivation de cette option vous permet de retenir le formatage précis des valeurs. Par exemple, cela peut être utile pour satisfaire une facette de pattern dans un schéma exigeant un nombre spécifique de chiffres décimaux dans une valeur numérique. Vous pouvez utiliser des fonctions de mappage pour formater le nombre en tant que string dans le format requis, puis mapper ce string dans la cible.

Veuillez noter que la désactivation de cette option désactivera la détection de valeurs invalides, par exemple écrire des lettres dans des champs numériques.

- Sortie Pretty print

Reformate le document XML de sortie pour lui donner un aspect structuré. Chaque nœud enfant est décalé de son parent par un seul caractère de tabulation.

- Créer signature numérique (*Enterprise Edition*)

  Vous permet d'ajouter une signature numérique dans le fichier d'instance de sortie XML. L'ajout d'une signature numérique est possible si vous sélectionnez Built-in en tant que langage de a transformation .

Encodage de sortie

Vous permet de spécifier les paramètres suivants du fichier d'instance de sortie :

- Nom d'encodage
- Tri d'octets
- Si le caractère byte order mark (BOM) doit être inclus

Par défaut, tout nouveau composant à l'encodage défini dans l'option *Encodage par défaut pour les nouveaux composants*. Vous pouvez accéder à cette option depuis **Outils | Options** (section *Généralités*).

Si le mappage génère XSLT 1.0/2.0, l'activation de la case à cocher *Byte Order Mark* n'a pas de conséquence**,** étant donné que ces langages ne prennent pas en charge Byte Order Marks.

Fichier de Feuille de style StyleVision Power

Cette option vous permet de sélectionner ou de créer un fichier de feuille de style Altova StyleVision. Ces fichiers vous permettent de sortir des données depuis le fichier d'instance XML à une variété de formats appropriés pour le reporting, comme HTML, RTF et autres. Voir aussi Utiliser des chemins relatifs sur un Composant 41.

Autres paramètres

- Activer les optimisations de traitement d'entrée basées sur min/maxOccurs

  Cette option vous permet une gestion spéciale pour les séquences connues pour contenir exactement un item, comme les attributs ou les éléments enfant requis avec `minOccurs` et `maxOccurs="1"`. Dans ce cas, le premier item de la séquence est extrait, puis l'item est traité directement en tant que valeur atomique (et pas en tant que séquence).

  Si les données d'entrée **ne sont pas valides** par rapport au schéma, une séquence vide peut survenir dans un mappage, ce qui interrompt le mappage et affiche un message d'erreur. Pour permettre le traitement d'une **entrée invalide**, désactivez cette case à cocher.

- Enregistrer tous les chemins de fichier relatifs au fichier MFD

  Lorsque cette option est activée, MapForce enregistre les chemins de fichier affichés dans le dialogue **Paramètres de composant** relatif à l'emplacement du fichier de MapForce Design (**`.mfd`**). Voir aussi Utiliser des chemins relatifs et absolus 41.

4.1.2 Types dérivés

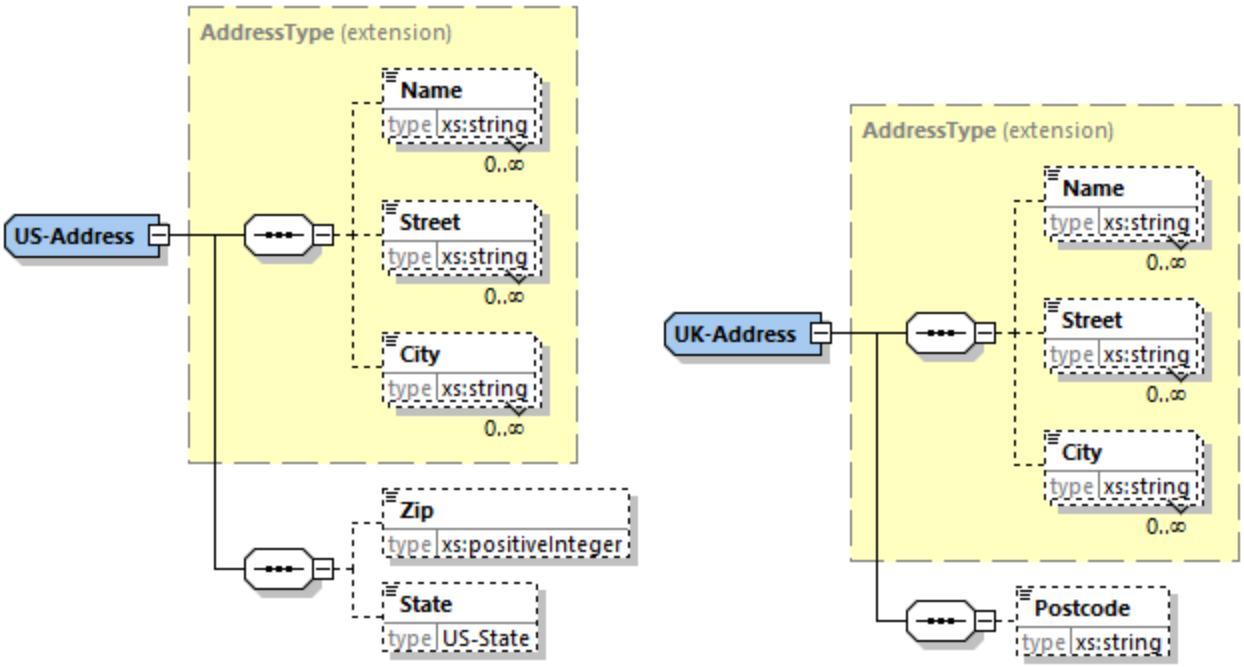
Cette rubrique explique comment utiliser les types dérivés dans les mappages. Les types dérivés sont définis dans la Spécification de schéma W3C XML (la section 2.5.2). Pour un bref aperçu des types dérivés et primitifs, voir la documentation Microsoft. Pour utiliser les types dérivés dans un mappage, vous devez spécifier l'attribut `xsi:type` dans votre fichier XML (par ex., `<Address xsi:type="UK-Address">`).

Scénarios possibles

Cette sous-section décrit un scénario possible d'utilisation d'un type dérivé. Par exemple, nous avons une entreprise avec deux filiales : une au Royaume-Uni et l'autre aux États-Unis d'Amérique. Maintenant, nous voulons avoir deux listes (`UKCustomers` et `USCustomers`), dont chacune inclura des informations sur l'adresse de la filiale respective et tous les clients associés à cette filiale.

Définition des types dérivés

Les captures d'écran ci-dessous illustrent la définition des types dérivés appelés `US-Address` et `UK-Address` (*XMLSpy mode Schéma*). Les éléments `UK-Address` et `UK-Address` ont le même type de base appelé `AddressType` qui inclut les éléments `Name`, `Street` et `City`. Dans l'élément `US-Address`, le type de base a été élargi pour inclure `Zip` et `State`, tandis que l'élément `UK-Address` inclut le type de base et l'élément `Postcode`. À des fins d'illustration, nous mapperons uniquement l'élément `UK-Address` vers le fichier cible.

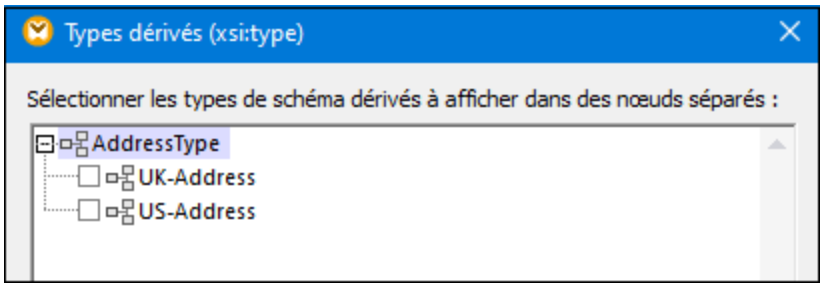


Type dérivé dans un mappage

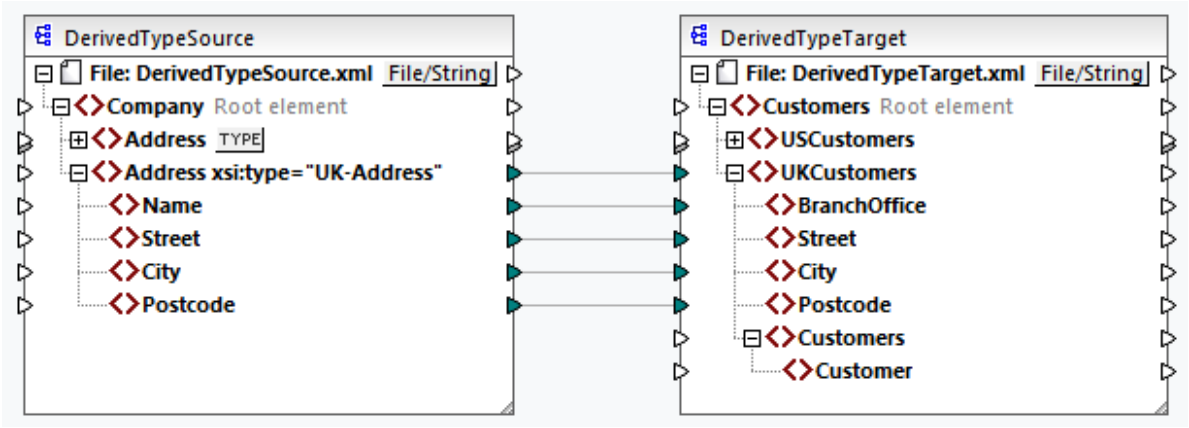
Les instructions ci-dessous montrent comment mapper les données depuis le type dérivé. Notre objectif est de mapper l'information sur les bureaux au UK vers l'élément `UKCustomers`. Les exemples de fichiers sont disponibles dans le dossier **`Tutoriel`**.

1. Dans le menu **Insérer**, cliquez sur **Schéma XML/Fichier** et ouvrez **`DerivedTypeSource.xml`**. Ce fichier XML est basé sur **`DerivedTypeSource.xsd`**.

2. Insérez le fichier cible appelé **`DerivedTypeTarget.xsd`**. Notez que le schéma cible ne doit pas inclure l'attribut `xsi:type`.
3. Cliquez sur la touche TYPE à côté de l'élément `Address` dans le composant source. Cette touche indique que les types dérivés existent pour cet élément dans le schéma.
4. La boîte de dialogue **Types dérivés** (*voir la capture d'écran ci-dessous*) vous permet de sélectionner tous types dérivés disponibles pour cet élément spécifique. Dans notre modèle de mappage, nous voulons que seule `UK-Address` soit mappée.



5. Dès que vous sélectionnez la case à cocher à côté du type dérivé `UK-Address`, un nouvel élément appelé `Address xsi:type="UK-Address"` apparaît dans le composant.
6. Maintenant, connectez les nœuds tel qu'affichés dans le mappage ci-dessous.



Sortie

Cliquez sur le volet **Sortie** vous montrera le résultat suivant :

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

Le modèle de mappage est enregistré sous **`Tutorial\DerivedType.mfd`**. Vous pouvez également ajouter un autre fichier XML source qui comprend des informations sur les clients au UK et mapper ces données au nœud `Customers` dans le composant cible. De cette manière, l'élément `UKCustomers` inclura l'information sur l'adresse UK et tous les clients associés à cette filiale.

4.1.3 Valeurs NULL

Cette section décrit comment MapForce gère les valeurs NULL dans les composants source et cible. Pour pouvoir utiliser l'attribut `xsi:nil="true"` dans votre fichier XML, vous devez spécifier l'attribut `nillable="true"` pour les élément(s) dans votre fichier de schéma. Pour en savoir plus sur les attributs `nillable` et `xsi:nil`, voir la spécification W3C. Notez que l'attribut `xsi:nil` n'est pas visible dans une arborescence de composant dans le volet **Mappage**.

Les sous-sections ci-dessous illustrent quelques scénarios possibles de valeurs NULL de mappage.

Valeurs NULL dans les composants XML

Cette sous-section décrit quelques uns des scénarios possibles de mappage des éléments avec un attribut `xsi:nil="true"`.

Uniquement l'élément source a xsi:nil="true"/Les deux éléments source et cible ont xsi:nil="true"
Ce scénario a les conditions suivantes :

- La connexion est target-driven[50].
- L'élément source a un attribut `xsi:nil="true"`. L'élément cible correspondant n'a pas cet attribut.
- En alternative, les deux éléments source et cible peuvent avoir les attributs `xsi:nil="true"`.
- Les attributs `nillable="true"` doivent être définis dans les schémas source et cible.
- Les éléments source et cible sont de type simple.

Dans ce cas, l'élément cible aura l'attribut `xsi:nil="true"` dans le fichier de sortie, tel qu'affiché dans le modèle de fichier de sortie ci-dessous (*en surbrillance jaune*).

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

Note : si l'attribut `nillable="true"` n'est *pas* défini dans le schéma cible, l'élément cible correspondant sera vide dans la sortie.

Seul l'élément cible a xsi:nil="true"
Ce scénario a les conditions suivantes :

- La connexion est target-driven[50].
- L'élément source n'a pas d'attribut `xsi:nil="true"`.
- L'élément cible correspondant a un attribut `xsi:nil="true"`.
- L'élément source et cible peut être de type simple ou complexe.

Dans ce cas, l'élément source écrasera l'élément cible contenant l'attribut `xsi:nil="true"`. L'exemple ci-dessous montre un modèle de fichier de sortie. L'élément `<genre>` inclut l'attribut `xsi:nil="true"` dans l'élément cible. Toutefois, cet élément a été écrasé lors de l'exécution de mappage. Pour cette raison, l'élément `<genre>` (*en surbrillance jaune*) a `Fiction` dans la sortie.

```
</publication>
```

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

L'élément source Complex-type/les éléments complex-type ont xsi:nil="true"
Ce scénario a les conditions suivantes :

- La connexion est target-driven[50].
- L'élément source est de type complexe. Dans notre exemple, l'élément source a un attribut `id="213"` et un attribut `xsi:nil="true"`. L'élément cible correspondant est de type complexe et a un attribut `id="124"` , mais n'a pas d'attribut `xsi:nil="true"`.
- En alternative, les éléments source et cible, dont les deux sont de type complexe, peuvent avoir des attributs `xsi:nil="true"`.

Dans ce cas, l'élément source écrasera l'élément cible (*en surbrillance jaune ci-dessous*. Toutefois, l'attribut `xsi:nil="true"` ne sera pas écrit dans le fichier de sortie automatiquement. Pour voir l'attribut `xsi:nil="true"` dans l'élément cible dans le fichier de sortie, utilisez la connexion copy-all[55].

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

Fonctions utiles

Les fonctions suivantes pourraient vous aider à vérifier, remplacer et assigner les valeurs NULL :

- **is-xsi-nil**[270] : Aide à vérifier explicitement si un élément source a un attribut `xsi:nil` définit à `true`.
- **substitute-missing**[304] : Substitue une valeur NULL dans l'élément source avec quelque chose de spécifique.
- **set-xsi-nil**[272] : Assigne l'attribut `xsi:nil="true »` à un élément cible. Ceci fonctionne pour les éléments cible de types simple et complexe.
- **substitute-missing-with-xsi-nil**[274] : S'il y a un contenu, il sera écrit à l'élément cible ; s'il y a des valeurs qui manquent, utilisez cette fonction résultera en élément cible avec un attribut `xsi:nil="true"` dans la sortie.
- Connecter une fonction **exists**[278] à un élément source avec une valeur NULL retourne `true`, même si l'élément n'a pas de contenu.

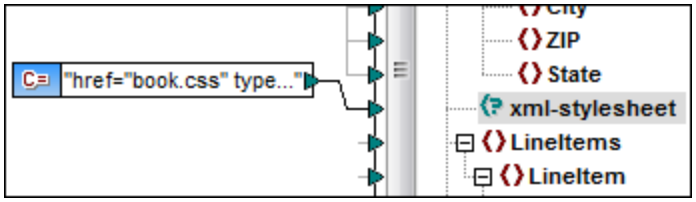
Veuillez noter que les fonctions qui génèrent `xsi:nil` ne peuvent pas être passées par les fonctions ou composants qui opèrent uniquement sur des valeurs (telles que la fonction **if-else**).

4.1.4 Commentaires et Instructions de traitement

Cette rubrique explique comment insérer des commentaires et instructions de traitement dans les composants XML cible. Notez que les commentaires et nœuds de l'instruction de traitement n'ont que des connexions d'entrée. Les commentaires et instructions de traitement ne peuvent pas être définis pour les nœuds qui font partie d'une connexion copy-all[55]. Les commentaires et instructions de traitement sont définies dans les Spécifications W3C.

Insérer un commentaire/instruction de traitement
Pour insérer une instruction de traitement ou un commentaire, suivez les étapes ci-dessous :

1. Double-cliquez sur un élément dans le composant et sélectionnez **Ajouter Commentaire/Instruction de traitement Avant/Après**. Lorsque vous insérez une instruction de traitement, vous allez aussi avoir besoin de saisir son nom. Dans cet exemple ci-dessous, une instruction de traitement appelée `xml-stylesheet` a été insérée après l'élément `State`.



3. Pour fournir la valeur d'un commentaire ou une instruction de traitement, vous pouvez utiliser une constante, par exemple (*voir la capture d'écran ci-dessus*).

Note: les instructions de traitement multiples peuvent être ajoutés avant ou après tout élément dans le composant de cible.

Note : seul un commentaire peut être ajouté avant et après un nœud de cible unique. Pour créer des commentaires multiples, utilisez la fonction d'entrée dupliquée[40].

Supprimer un commentaire/instruction de traitement
Pour supprimer un commentaire/instruction de traitement, double-cliquez sur le nœud respectif, sélectionnez **Commentaire/Instruction de traitement**, puis sélectionnez **Supprimer Commentaire/Instruction de traitement** depuis le menu contextuel.

4.1.5 Sections CDATA

Les sections CDATA sont utilisées pour échapper des blocs de texte contenant des caractères qui devraient normalement être interprétés en tant que balise. Pour plus d'informations sur les sections de CDATA, voir la Spécification W3C. Les nœuds cible recevant des données comme sections CDATA peuvent être comme suit : Données XML, données XML intégrées dans les champs de base de données et éléments enfant XML de dimensions typées dans une cible XBRL. Les sections CDATA peuvent aussi être définies sur des nœuds dupliqués et des nœuds `xsi:type`.

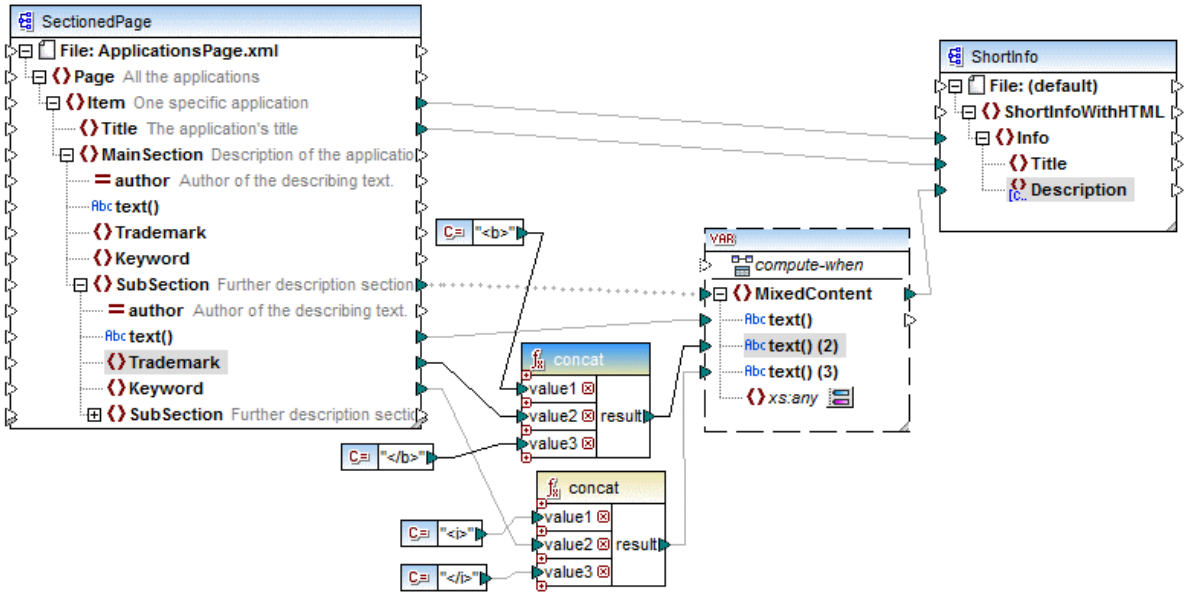
Pour créer une section CDATA, double-cliquez sur le nœud cible pertinent et sélectionnez **Écrire contenu comme section CDATA**. Une invitation apparaît vous avertissant que les données d'entrée ne doivent pas

contenir le délimiteur de fermeture de la section CDATA `]]>`. L'icône `[C..` apparaît en-dessous de l'onglet élément, qui indique que ce nœud cible est maintenant défini comme section CDATA.

Exemple

Cet exemple ci-dessous montre un scénario dans lequel la section CDATA pourrait être utile. L'exemple de mappage appelé **`MapForceExamples\HTMLinCDATA.mfd`** (*voir la capture d'écran ci-dessous*) a les aspects suivants :

- L'élément `SubSection` a du contenu mixte. Pour plus d'information sur les nœuds de contenu mixte, voir Connexions orientées vers la source 50.
- Avec l'aide de la fonction **`concat`**, le contenu de l'élément `Trademark` aura les balises `<b></b>`.
- Le contenu de l'élément `Keyword` aura des balises `<i></i>`.
- Les données avec les nouvelles balises sont transmises aux nœuds `text()` dupliqués dans le même ordre que dans le document source.
- La sortie du nœud `MixedContent` est ensuite transmise dans le nœud `Description` dans le composant cible `ShortInfo`. Le nœud `Description` a été défini comme section CDATA.



Sortie

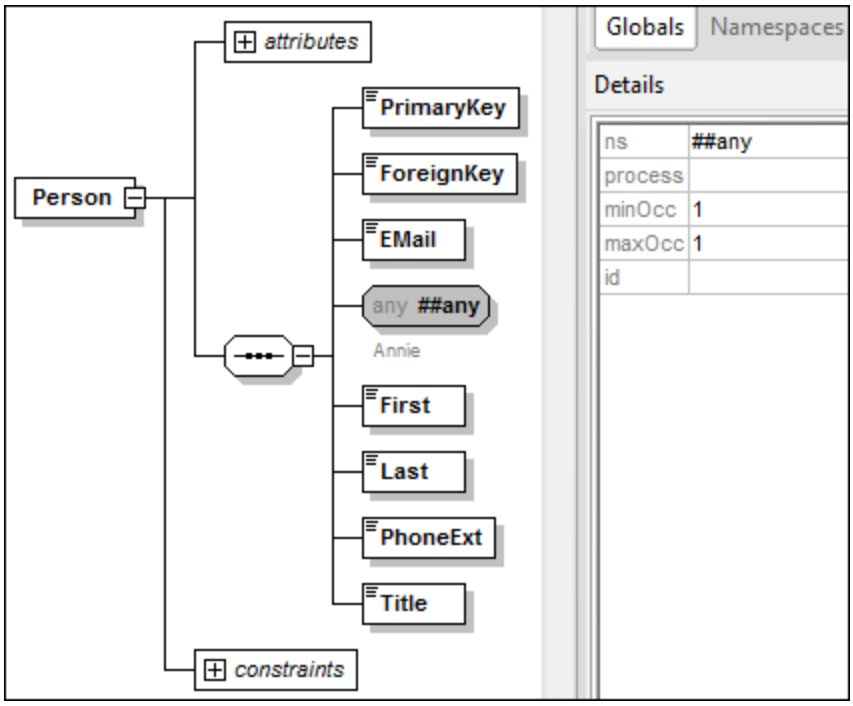
Cliquez dans le volet **Sortie** pour consulter la section CDATA dans le nœud `Description` (*voir la capture d'écran ci-dessous*).

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

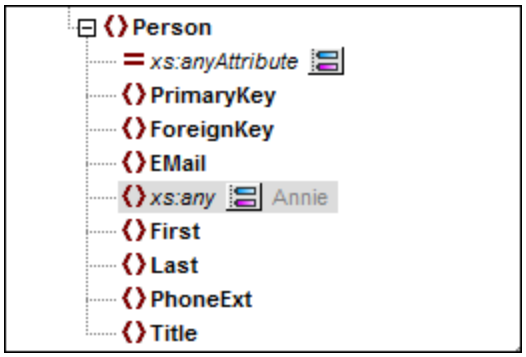
4.1.6 Caractères génériques - xs:any / xs:anyAttribute

Cette rubrique explique comment gérer les caractères génériques dans les mappages. Les caractères génériques `xs:any` (et `xs:anyAttribute`) vous permettent d'utiliser tout élément/attribut depuis les schémas. Pour plus d'informations sur les caractères génériques, voir la Spécification W3C.

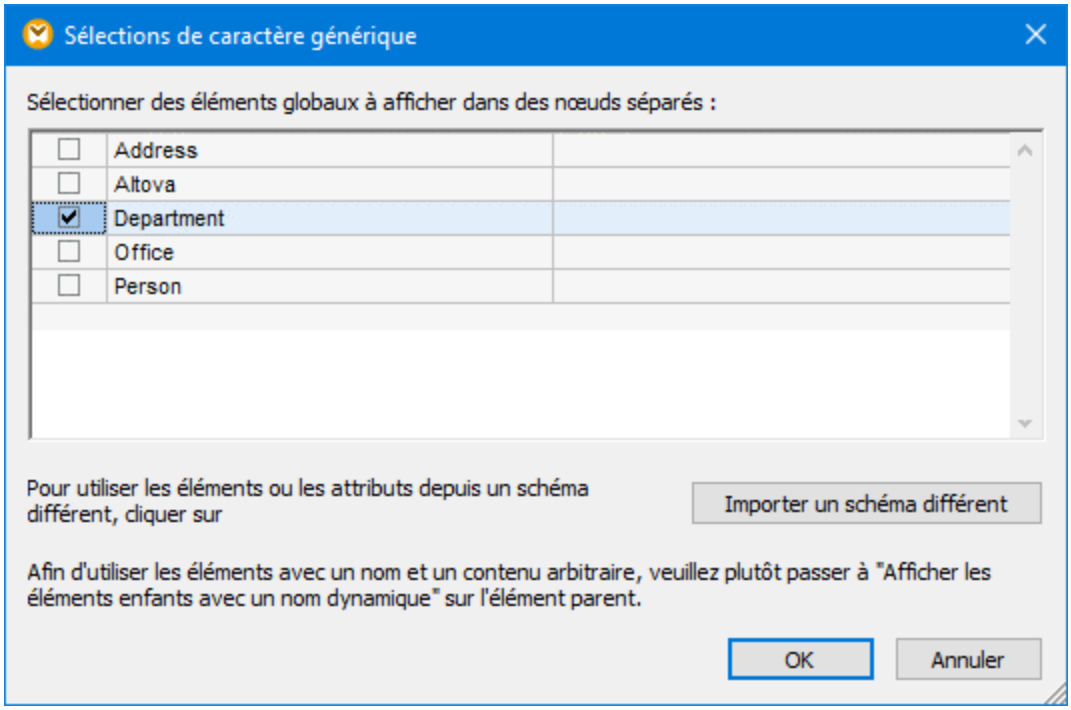
Caractères génériques dans la définition de schéma
La capture d'écran ci-dessous affiche qu'un élément `xs:any` a été défini comme élément enfant de l'élément `Person` (*mode Schéma dans Altova XMLSpy*).



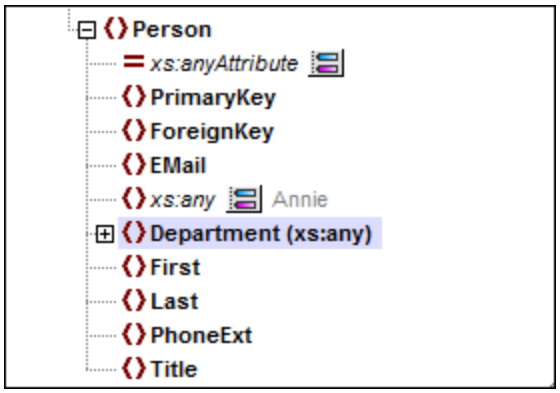
Caractères génériques dans MapForce.
Lorsqu'un caractère générique est défini pour un élément et/ou attribut, ce nœud aura un bouton (**Changer Sélection**) à côté de celui-lui (*voir la capture d'écran ci-dessous*).



Sélection de caractère générique
Désormais, notre objectif est d'ajouter un autre élément comme nœud séparé. Cliquez sur le bouton pour voir la liste des éléments que vous pouvez ajouter à l'arborescence. Notez que seuls les éléments et attributs globalement déclarés dans votre schéma peuvent être vus dans la boîte de dialogue **sélections de caractère générique**(*voir la capture d'écran ci-dessous*).



Pour notre exemple, nous avons choisi `Département`. Notez que les éléments de caractère générique et les attributs sont insérés après le nœud avec le bouton . Maintenant, nos composants ont l'air comme suit :



Vous pouvez maintenant mapper de/vers ces nœuds comme d'habitude. Dans un composant, les éléments de caractère générique et les attributs sont marqués avec `(xs:any)` et (`xs:anyAttribute`), respectivement (*voir la capture d'écran ci-dessus*).

Supprimer les caractères génériques
Pour supprimer un nœud de caractère générique, cliquez sur le bouton et la case à cocher correspondante dans la boîte de dialogue **sélections de caractère générique**.

Éléments/attributs d'un schéma différent

La boîte de dialogue **sélections de caractère générique** (*voir ci-dessus*) vous permet d'utiliser les éléments/attributs d'un schéma différent. Cliquer sur le bouton **Importer un schéma différent** vous donnera les options suivantes : (i) importer un fichier de schéma et (ii) générer un schéma wrapper (*voir la description ci-dessous*).

Importer un schéma
L'option **Importer schéma** importe le schéma externe dans le schéma actuel attribué au composant. Veuillez noter que cette option écrase le schéma existant du composant sur le disque. Si le schéma actuel est un schéma à distance qui a été ouvert depuis une URL (voir Ajouter des composants depuis une URL 36) et non depuis le disque, il ne peut pas être modifié. Dans ce cas, utilisez l'option **Générer schéma wrapper**.

Générer un schéma wrapper
L'option **Générer schéma wrapper** crée un nouveau fichier de schéma appelé un *schéma wrapper*. L'avantage de l'utilisation de cette option est que le schéma existant du composant n'est pas modifié. Au lieu, un nouveau schéma sera créé qui comprendra aussi bien le schéma existant que le schéma à importer. Si vous choisissez cette option, vous serez invité à choisir où le schéma wrapper devra être enregistré. Par défaut, le schéma de wrapper a le format suivant : **`somefile-wrapper.xsd`**.

Une fois avoir enregistré le schéma wrapper, celui-ci sera attribué automatiquement par défaut au composant. MapForce vous demandera également si vous voulez ajuster l'emplacement du schéma pour que vous puissiez référencer le schéma principal précédent. Cliquez sur **Oui** pour passer au schéma précédent ; sinon, cliquez sur **Non** pour garder l'attribution du schéma wrapper récemment créé au composant.

Caractères génériques et noms de nœud dynamiques

Néanmoins, il peut se produire des situations dans lesquelles les éléments et/ou les attributs dans une instance sont trop nombreux pour être déclarés dans le schéma. Considérez le fichier exemple suivant :

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

Pour de telles situations, il est recommandé d'utiliser les noms dynamiques de nœuds à la place des caractères génériques. Pour plus d'information, voir Nom du nœud stratégies de mappage 383.

4.1.7 Espaces de noms personnalisés

Lorsqu'un mappage produit une sortie XML, MapForce dérive automatiquement l'espace de nom (ou définit des espaces de noms) de chaque élément et attribut depuis le schéma cible. Il s'agit du comportement par défaut approprié pour la plupart des scénarios de la génération de sortie XML. Toutefois, il existe des cas pour lesquels vous voulez déclarer manuellement l'espace de nom d'un élément directement depuis le mappage.

Déclarer des espaces de noms personnalisés est uniquement pertinent pour les composants XML cible, et s'applique uniquement aux éléments. La commande **Add Namespace** n'est pas disponible pour les attributs, nœuds du caractère générique et pour les nœuds qui reçoivent les données de la connexion copy-all .

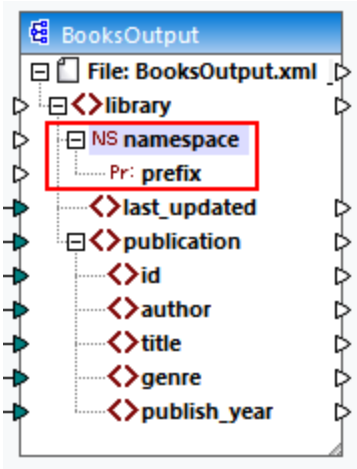
Pour comprendre comment les espaces de noms personnalisés fonctionnent, suivez les instructions dans la sous-section ci-dessous.

Déclarer l'espace de nom manuellement

Pour cet exemple, vous allez avoir besoin du mappage suivant : **`BasicTutorials\Tut1-SchemaToSchema.mfd`**.

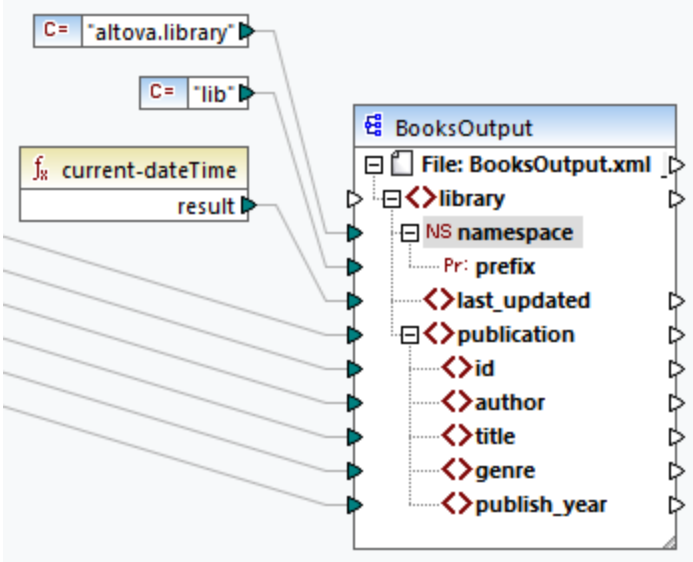
Ajouter un espace de nom

Ouvrez le mappage, cliquez avec la touche de droite sur le nœud `bibliothèque` dans le composant `BooksOutput` et sélectionnez **Add Namespace** depuis le menu contextuel. Désormais, deux nouveaux nœuds sont disponibles dans l'élément `bibliothèque` : `namespace` et `prefix` (*voir la capture d'écran ci-dessous*).



Fournir les valeurs d'espace de noms

La prochaine étape est de fournir les valeurs aux nœuds `namespace` et `prefix`. À cette fin, nous utiliserons les deux constantes avec les valeurs de string suivantes : `altova.library` et `lib` (*voir la capture d'écran ci-dessous*).



Note : les deux connecteurs d'entrée `namespace` et `prefix` doivent être mappés, même si vous leur fournissez des valeurs vides.

Sortie
Dans la sortie générée, un attribut `xmlns:<prefix>="<namespace>"` est ajouté à l'élément, où `<prefix>` et `<namespace>` sont des valeurs qui proviennent du mappage. La sortie générée ressemblera à l'exemple suivant(*notez la partie en surbrillance*) :

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

Vous pouvez aussi déclarer plusieurs espaces de noms pour le même élément, si nécessaire. À cette fin, cliquez une nouvelle fois sur la touche de droite dans le nœud et sélectionnez **Ajouter espace de noms** depuis le menu contextuel. Une nouvelle paire de nœuds d'espace de noms et de préfixe deviennent disponibles, auxquels vous pouvez connecter les valeurs du nouveau préfixe et de l'espace de noms.

Déclarer un espace de nom par défaut
Si vous voulez déclarer un espace de noms par défaut, mappez une valeur de string vide à `prefix`. La sortie ressemblera alors à l'exemple suivant (*notez la partie en surbrillance*):

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

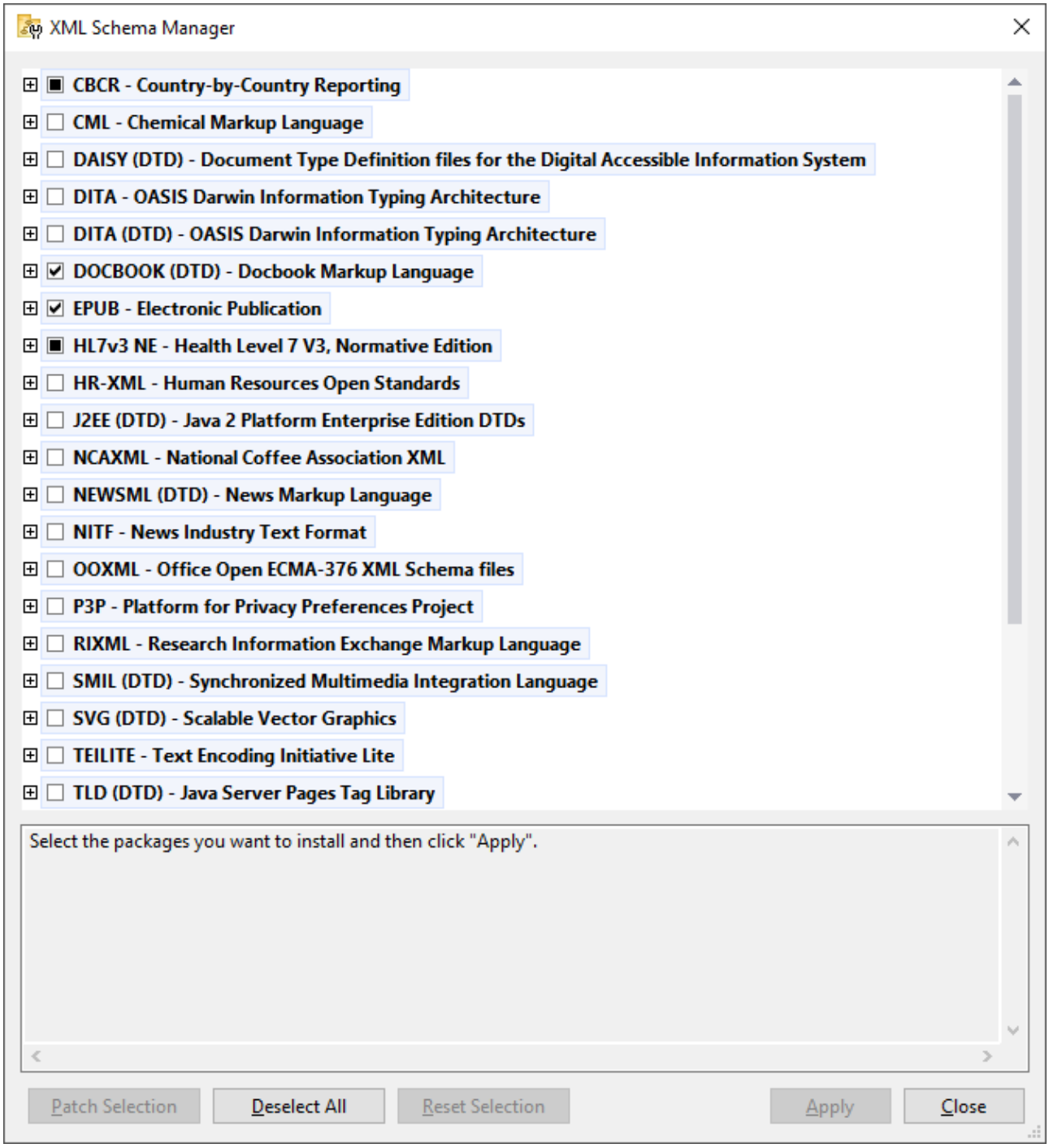
Si vous souhaitez créer des préfixes pour les noms d'attribut, par exemple `<number prod:id="prod557">557</number>`, vous pouvez le faire soit en utilisant l'accès dynamique aux attributs d'un nœud (voir Mapper des noms de nœud 383), soit en éditant le schéma de manière à ce qu'il ait un attribut `prod:id` pour `<number>`.

Supprimer l'espace de noms
Pour supprimer une déclaration d'espace de noms précédemment ajoutée, cliquer avec la touche de droite dans le nœud `ns:namespace` et choisissez **Supprimer espace de noms** du menu contextuel.

4.1.8 Gestionnaire de schéma

XML Gestionnaire de schéma est un outil qui propose un moyen centralisé d'installer et de gérer des schémas XML (DTD pour XML et Schémas XML) pour une utilisation sur toutes les applications « XML-Schema-aware » d'Altova, y compris MapForce

- Sur Windows, Gestionnaire de schéma a une interface utilisateur graphique *(voir la capture d'écran ci-dessous*) et est aussi disponible dans la ligne de commande. (Les applications desktop d'Altova sont disponibles sur Windows uniquement ; *voir la liste ci-dessous*.)
- Sur Linux et Mac Gestionnaire de schéma, l'outil est disponible uniquement dans la ligne de commande. (Les applications serveur d'Altova sont disponibles sur Windows, Linux et macOS ; *voir la liste ci-dessous*.)



Application d'Altova qui fonctionnent avec Schema Manager

Applications desktop (Windows uniquement)	Applications de serveur (Windows, Linux, macOS)
XMLSpy (toutes éditions)	RaptorXML Server, RaptorXML+XBRL Server

MapForce (toutes éditions)	StyleVision Server
StyleVision (toutes éditions)	
Authentic Desktop Enterprise Edition	

Installation et désinstallation de Gestionnaire de schéma

Gestionnaire de schéma est installé automatiquement quand vous installez d'abord une nouvelle version de l'Altova Mission Kit ou toute application « XML-schema-aware » d'Altova (*voir la table ci-dessus*).

De même, il est supprimé automatiquement lorsque vous désinstallez la dernière application XML-schema-aware d'Altova depuis votre ordinateur.

Fonctions <% SCHEMA-MANAGER%>

Gestionnaire de schéma propose les fonctions suivantes :

- Affiche les schémas XML installés sur votre ordinateur et contrôle si de nouvelles versions sont disponibles pour le téléchargement.
- Télécharge des versions plus récentes des schémas XML indépendamment du cycle de release des produits Altova. (Altova stocke des schémas en ligne et vous pouvez les télécharger via Gestionnaire de schéma.)
- Installer ou désinstaller une des multiples versions d'un schéma donné (ou toutes les versions, si nécessaire).
- Un schéma XML peut avoir des dépendances sur d'autres schémas. Lorsque vous installez ou désinstallez un schéma particulier, Gestionnaire de schéma vous informe sur d'autres schémas dépendants et les installera ou désinstallera également automatiquement.
- Gestionnaire de schéma utilise le mécanisme du catalogue XML pour mapper les références de schéma aux fichiers locaux. Dans le cas de larges schémas XML, le traitement sera plus rapide que si les schémas étaient à un emplacement à distance.
- Tous les schémas majeurs sont disponibles via Gestionnaire de schéma et sont régulièrement mis à jour pour les dernières versions. Ceci vous fournit une ressource unique pour gérer tous vos schémas et les mettre à disposition de toutes les applications « XML-schema-aware » d'Altova.
- Les changements réalisés dans Gestionnaire de schéma prennent effet pour tous les produits d'Altova sur cet appareil.

Comment cela fonctionne ?

Altova stocke tous les schémas XML utilisés dans les produits Altova en ligne. Ce référentiel est mis à jour lorsque de nouvelles versions de schémas sont publiées. Gestionnaire de schéma affiche des informations sur les derniers schémas disponibles lorsqu'ils sont appelés dans son formulaire GUI de même que sur CLI. Vous pouvez ensuite installer, mettre à jour ou désinstaller les schémas via Gestionnaire de schéma.

Gestionnaire de schéma installe également les schémas d'une autre manière. Sur le site web d'Altova (https://www.altova.com/schema-manager), vous pouvez sélectionner un schéma et ses Schémas dépendants que vous souhaitez installer. Le site web préparera un fichier de type `.altova_xmlschemas` pour le téléchargement qui contient des informations sur la sélection de schéma. Lorsque vous double-cliquez sur ce fichier ou le passez à Gestionnaire de schéma via CLI comme argument de la commande `installer`[141], Gestionnaire de schéma installera les schémas que vous avez sélectionnés.

Cache local : suivre vos schémas
Toutes les informations sur les schémas installés sont suivies dans un répertoire cache centralisé sur votre ordinateur, situé ici :

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

Ce répertoire cache est mis à jour régulièrement avec le dernier statut des schémas dans l'emplacement de stockage en ligne d'Altova. Ces mises à jour sont réalisées aux moments suivants :

- À chaque fois que vous lancez Gestionnaire de schéma.
- Lorsque vous exécutez MapForce pour la première fois dans un jour donné du calendrier.
- Si MapForce est ouvert plus de 24 heures, le cache est mis à jour toutes les 24 heures.
- Vous pouvez aussi mettre à jour le cache en exécutant la commande de `mise à jour`[144] dans l'interface de ligne de commande.

Pour cette raison, le cache permet à Gestionnaire de schéma de suivre continuellement vos schémas installés par rapport aux schémas disponibles en ligne sur le site web d'Altova.

Ne modifiez pas le cache manuellement !

Le répertoire de cache local est entretenu automatiquement sur la base des schémas que vous installez ou désinstallez. Il ne devrait pas être altéré ou supprimé manuellement. Si vous êtes amené à réinitialiser Gestionnaire de schéma à son état original "intact", alors, sur l'interface de la ligne de commande (CLI) : (i) exécutez la commande `reset`[143], et (ii) exécutez la commande `initialize`[140]. (En alternative, exécutez la commande **`reset`** avec l'option **`--i`**.)

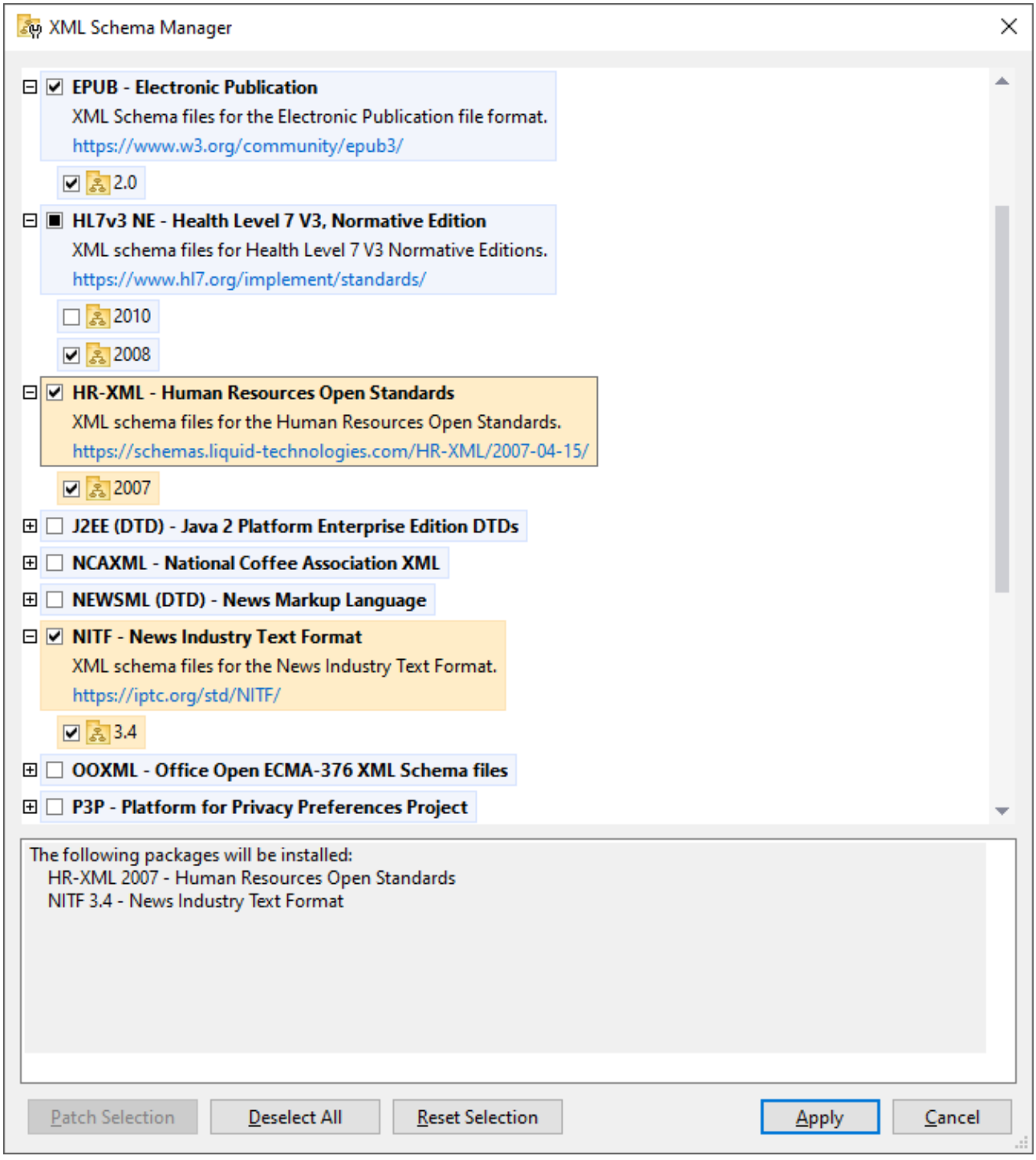
4.1.8.1 Exécuter Schema Manager

Interface utilisateur graphique

Vous pouvez accéder à la GUI de Gestionnaire de schéma des manières suivantes :

- *Durant l'installation de MapForce:* Vers la fin de la procédure d'installation, sélectionnez la case à cocher *Invoke Altova XML-Schema Manager* pour accéder directement à la GUI de Gestionnaire de schéma. Ceci vous permettra d'installer les schémas au cours de la procédure d'installation de votre application Altova.
- *Après l'installation de MapForce:* Une fois que votre installation a été installée, vous pouvez accéder à la GUI de Gestionnaire de schéma à tout moment, via la commande de menu **Tools | XML Schema Manager**.
- Via le fichier **`.altova_taxonomies`** téléchargé de Altova website: Double-cliquez sur le fichier téléchargé pour exécuter la GUI de Gestionnaire de schéma, qui sera configurée pour installer les schémas que vous avez sélectionnés (le site web) pour installation.

Une fois que la GUI de Gestionnaire de schéma (*capture d'écran ci-dessous*) a été ouverte, les schémas déjà installés seront affichés tels sélectionnés. Si vous voulez installer un schéma additionnel, sélectionnez-le. Si vous voulez désinstaller un schéma déjà installé, désélectionnez-le. Une fois que vous avez faits vos sélections et/ou désélections, vous êtes prêts pour appliquer vos changements. Les schémas qui seront installés ou désinstallés seront mis en surbrillance et un message sur les modifications à venir sera posté dans le volet Messages au niveau inférieur de la fenêtre Gestionnaire de schéma (*voir la capture d'écran*).



Interface de ligne de commande

Vous pouvez exécuter Gestionnaire de schéma depuis une interface de ligne de commande en sélectionnant son fichier exécutable, **`xmlschemamanager.exe`**.

Le fichier **`xmlschemamanager.exe`** est situé dans le dossier suivant :

- *Sur Windows :* `C:\ProgramData\Altova\SharedBetweenVersions`
- *Sur Linux ou macOS (application serveur uniquement) :* `%INSTALLDIR%/bin`, où `%INSTALLDIR%` est le répertoire d'installation du programme.

Vous pouvez alors utiliser toute commande dans la section de référence de la commande CLI[139].

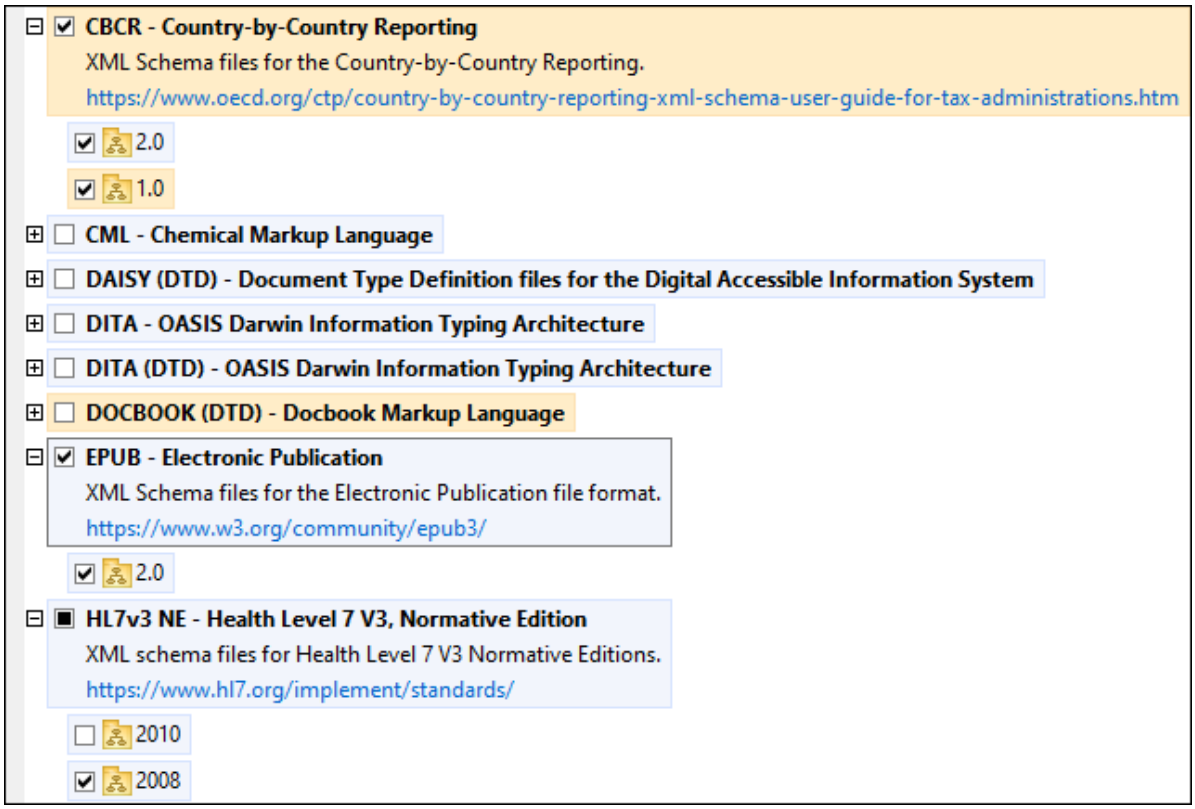
Pour afficher l'aide pour la commande, exécutez l'étape suivante :

- *Sur Windows :* `xmlschemamanager.exe --help`
- *Sur Linux ou macOS (application serveur uniquement) :* `sudo ./xmlschemamanager --help`

4.1.8.2 Catégories de statut

Gestionnaire de schéma catégorise les schémas sous sa gestion comme suit :

- *Schémas installés.* Ceux-ci sont affichés dans la GUI avec leurs cases à cocher sélectionnées (*dans la capture d'écran ci-dessous, les versions cochées et bleues des schémas EPUB et HL7v3 NE sont des schémas installés*). Si toutes les versions de schéma sont sélectionnées, alors la marque de sélection est une coche. Si au moins une version de schéma est décochée, alors la coche de sélection est un carré coloré plein. Vous pouvez décocher un schéma installé pour le **désinstaller** ; (*dans la capture d'écran ci-dessous, le DocBook DTD est installé et a été désélectionné, le préparant ainsi pour la désinstallation*).
- *Schémas désinstallés disponibles.* Ils sont affichés dans la GUI avec leurs cases à cocher non sélectionnées. Vous pouvez sélectionner les schémas que vous souhaitez **installer**.



- Les *Schémas pouvant être mises à niveau* sont ceux qui ont été revus par leurs émetteurs depuis qu'ils ont été installés. Ils sont indiqués dans la GUI par une icône  . Vous pouvez **retoucher** le schéma installé avec la révision disponible.

Points à noter

- Dans la capture d'écran ci-dessus, les deux schémas CBCR sont cochés. Celui avec un arrière-plan bleu est déjà installé. Celui avec un arrière-plan jaune est désinstallé et a été sélectionné pour l'installation. Notez que le schéma HL7v3 NE 2010 n'est pas installé et n'a pas été sélectionné pour l'installation.
- Un arrière-plan jaune signifie que le schéma sera modifié d'une manière ou d'une autre quand le bouton **Appliquer** est cliqué. Si un schéma est décoché et a un arrière-plan jaune, cela signifie qu'il sera désinstallé quand le bouton **Appliquer** est cliqué. Dans la capture d'écran ci-dessus, le DocBook DTD a un tel statut.
- Lorsque vous exécutez Gestionnaire de schéma depuis la ligne de commande, la commande `list` 142 est utilisée avec différentes options pour pour recenser les différentes catégories de schémas :

`xmlschemamanager.exe list`	Recense tous les schémas installés et disponibles ; ceux pouvant être mis à niveau sont également indiqués.
`xmlschemamanager.exe list -i`	Recense les schémas installés uniquement ; ceux pouvant être mis à niveau sont également indiqués
`xmlschemamanager.exe list -u`	Recense les schémas pouvant être mis à niveau

Note : Sur Linux et macOS, **`use sudo ./ xmlschemamanager list`**

4.1.8.3 Retoucher ou Installer un schéma

Retoucher un schéma installé

Occasionnellement, des schémas XML peuvent recevoir des patchs (mises à niveau ou révisions) de leurs émetteurs. Lorsque Gestionnaire de schéma détecte que des patchs sont disponibles, ceux-ci sont indiqués dans les listes de schéma de Gestionnaire de schéma et vous pouvez installer les patchs rapidement.

Dans la GUI

Les patchs sont indiqués par l'icône . (*Voir aussi la rubrique précédente sur les catégories de statut*[135].) Si les patchs sont disponibles, le bouton **Patch Selection** sera activé. Cliquez dessus pour sélectionner et préparer tous les patchs pour installation. Dans la GUI, l'icône de chaque schéma sera patchée de à , et le volet des Messages en bas du dialogue recense les patchs qui doivent être appliqués. Lorsque vous êtes prêt pour installer des patchs sélectionnés, cliquez sur **Appliquer**. Tous les correctifs seront appliqués ensemble. Notez que si vous décochez un schéma marqué pour une correction, vous désinstallerez de fait ce schéma.

Sur le CLI

Pour appliquer un patch dans l'interface de ligne de commande :

1. Exécuter la commande `list -u`[142] . Cela liste tout schéma lorsque des mises à niveau sont disponibles.
2. Exécutez la commande `upgrade`[145] pour installer les patchs.

Installer un schéma disponible

Vous pouvez installer des schémas en utilisant soit la GUI Gestionnaire de schéma ou en envoyant à Gestionnaire de schéma les instructions d'install via la ligne de commande.

Note : Si le schéma actuel référence d'autres schémas, les schémas référencées sont aussi installés.

Dans la GUI

Pour installer des schémas utilisant la GUI Gestionnaire de schéma GUI, sélectionnez les schémas que vous voulez installer et cliquez sur **Appliquer**.

Vous pouvez aussi sélectionner les schémas que vous voulez installer sur le site web d'Altova et générer un fichier téléchargeable **`.altova_xmlschemas`**. Lorsque vous double-cliquez sur ce fichier, il ouvrira Gestionnaire de schéma avec les schémas que vous vouliez présélectionner. La seule chose qui vous reste à faire, c'est cliquer sur **Appliquer**.

Sur le CLI

Pour installer des schémas via la ligne de commande, exécutez la commande `install`[141] :

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

où **Schéma** est le schéma (ou les schémas) que vous voulez installer ou un fichier **.altova_xmlschemas**. Un schéma est référencé par un identifiant de format **<name>-<version>**. (Les identifiants de schémas sont affichés quand vous exécutez la commande list[142].) Vous pouvez saisir autant de schémas que vous le souhaitez. Pour plus de détails, voir la description de la commande install[141].

Note : sur Linux ou macOS, utilisez la commande **sudo ./xmlschemamanager**.

Installer un schéma requis
Lorsque vous exécutez une commande activée par XML dans MapForce, et que MapForce découvre qu'un schéma dont il a besoin pour exécuter la commande n'est pas présente ou est incomplète, Gestionnaire de schéma affichera l'information sur les schémas manquants. Vous pouvez ensuite installer directement tout schéma manquant via Gestionnaire de schéma.

Dans la GUI de Schema Manager, vous pouvez consulter tous les schémas précédemment installés à tout moment en exécutant Gestionnaire de schéma depuis **Outils | Gestionnaire de schéma**.

4.1.8.4 Désinstaller un schéma, Réinitialiser

Désinstaller un schéma

Vous pouvez désinstaller des schémas en utilisant soit la GUI Gestionnaire de schéma ou en envoyant à Gestionnaire de schéma les instructions d'installation via la ligne de commande.

Note : si la Police que vous voulez désinstaller référence d'autres Schémas, alors les Schémas référencées sont également désinstallées.

Dans la GUI
Pour désinstaller les schémas utilisant la GUI Gestionnaire de schéma, effacez leurs cases à cocher et cliquez sur **Appliquer**. Les schémas sélectionnés et leurs schémas référencées seront désinstallés.

Pour désinstaller tous les schémas, cliquez sur **Désélectionner tout** et cliquez sur **Appliquer**.

Sur le CLI
Pour désinstaller des schémas via la ligne de commande, exécutez la commande désinstaller[143] :

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

où chaque argument **Schéma** est le schéma que vous voulez désinstaller ou un fichier **.altova_xmlschemas**. Un schéma est spécifié par un identifiant qui a un format **<name>-<version>**. (Les identifiants de schémas sont affichés quand vous exécutez la commande list[142].) Vous pouvez saisir autant de schémas que vous le souhaitez. Pour plus de détails, voir la description de la commande désinstaller[143].

Note : sur Linux ou macOS, utilisez la commande **sudo ./xmlschemamanager**.

Réinitialiser Gestionnaire de schéma

Vous pouvez réinitialiser Gestionnaire de schéma. Ceci supprime toutes les schémas installés et le répertoire de mise sous cache.

- Dans la GUI, cliquez sur **Reset Selection**.
- Dans la CLI, exécutez la commande `reset`[143].

Une fois avoir exécuté cette commande, vous devrez exécuter la commande `initialize`[140], pour pouvoir recréer le répertoire de mise sous cache. En alternative, exécutez la commande `reset`[143] avec l'option `-i`.

Notez que `reset -i`[143] restaure l'installation originale du produit, il est recommandé d'exécuter la commande `update`[144] après avoir réalisé la réinitialisation. En alternative, exécutez la commande `reset`[143] avec les options `-i` and `-u`.

4.1.8.5 Interface de ligne de commande (CLI)

Pour appeler Gestionnaire de schéma dans la ligne de commande, vous devez connaître le chemin de l'exécutable. Par défaut, l'exécutable Gestionnaire de schéma est installé dans le chemin suivant :

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

Note : sur les systèmes Linux et macOS, une fois que vous avez changé le répertoire à celui contenant l'exécutable, vous pouvez appeler l'exécutable avec **`sudo ./xmlschemamanager`**. Le préfixe **`./`** indique que l'exécutable est le répertoire actuel. Le préfixe **`sudo`** indique que la commande que la commande doit être exécutée avec des privilèges root.

Syntaxe de ligne de commande

La syntaxe générale pour utiliser la ligne de commande est la suivante :

```
<exec> -h | --help | --version | <command> [options] [arguments]
```

Dans l'extrait ci-dessus, la barre verticale **`|`** sépare un ensemble d'items mutuellement exclusifs. Les crochets **`[]`** indiquent des items optionnels. De manière générale, vous pouvez saisir le chemin d'exécutable suivi soit par les options `--h`, `--help`, ou `--version` ou par une commande. Chaque commande peut contenir des options et des arguments. La liste des commandes est décrite dans les sections suivantes.

4.1.8.5.1 help

Cette commande propose une aide contextuelle pour les commandes liées à l'exécutable Gestionnaire de schéma.

Syntaxe

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

Où `[command]` est un argument optionnel qui spécifie un nom de commande valide.

Veuillez noter les points suivants :

- Vous pouvez invoquer de l'aide en saisissant la commande suivie par **`-h`** ou **`--help`**, par exemple : **`<exec> list-h`**
- Si vous tapez **`-h`** or **`--help`** directement après la commande exécutable et avant une commande, vous recevrez une aide générale (pas d'aide pour la commande), par exemple : **`<exec> -h list`**

Exemple

La commande suivante affiche une aide concernant la commande `list` :

```
xmlschemamanager help list
```

4.1.8.5.2 info

Cette commande affiche des informations détaillées pour chacun des schémas fournis en tant qu'argument de `Schéma`. Cette information inclut le titre, la version, description, l'éditeur et tout schéma soumis et tout schéma référencé, et informe si le schéma a été installé ou non.

Syntaxe

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

- L'argument **`Schéma`** est le nom d'un schéma ou une partie du nom de schéma. (Pour afficher une ID de pack de schéma et des informations détaillées sur son statut d'installation, vous devriez utiliser la commande `list`[142].)
- Utiliser **`<exec> info -h`** pour afficher l'aide de la commande.

Exemple

La commande suivante affiche l'information sur les derniers schémas **`DocBook-DTD`** et **`NITF`** :

```
xmlschemamanager info doc nitf
```

4.1.8.5.3 initialize

Cette commande initialise l'environnement Gestionnaire de schéma. Elle crée un répertoire de cache où les informations concernant tous les schémas sont stockées localement. L'initialisation est réalisée automatiquement la première fois qu'une application schema-cognizant d'Altova est installée. Vous n'aurez pas besoin d'exécuter cette commande dans des circonstances normales, mais vous devrez l'exécuter généralement après la commande **`reset`**.

Syntaxe

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

Options
La commande **initialize** accepte les options suivantes :

`--silent, --s`	Afficher uniquement des messages d'erreur. Le réglage par défaut est **faux**.
`--verbose, --v`	Afficher des informations détaillées lors de l'exécution. Le réglage par défaut est **faux**.
`--help, --h`	Afficher l'aide pour la commande.

Exemple
La commande suivante initialise Gestionnaire de schéma:

```
xmlschemamanager initialize
```

4.1.8.5.4 install

Cette commande installe un ou plusieurs schémas.

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

Pour installer de multiples schémas, ajoutez l'argument **Schéma** de nombreuses fois.

L'argument **Schéma** est l'un des suivants :

- Un identifiant de schéma (avoir un format de **<name>-<version>**, par exemple : **cbcr-2.0**). Pour trouver les identifiants de schémas que vous voulez, exécutez la commande `list`[142]. Vous pouvez aussi utiliser des identifiants abrégés s'ils sont uniques, par exemple **docbook**. Si vous utilisez un identifiant abrégé, alors la dernière version de ce schéma sera installée.
- Le chemin vers un fichier **.altova_xmlschemas** téléchargé depuis le site web d'Altova. Pour information sur ces fichiers, voir *Introduction à SchemaManager : Comment cela fonctionne-t-il ?*[129].

Options
La commande **install** accepte les options suivantes :

`--silent, --s`	Afficher uniquement des messages d'erreur. Le réglage par défaut est **faux**.
`--verbose, --v`	Afficher des informations détaillées lors de l'exécution. Le réglage par défaut est **faux**.
`--help, --h`	Afficher l'aide pour la commande.

Exemple

La commande suivante installe le schéma CBCR 2.0 (Country-By-Country Reporting) et le dernier DocBook DTD:

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

4.1.8.5.5 list

Cette commande recense les schémas sous la gestion de Gestionnaire de schéma. La liste affiche comme suit

- Tous les schémas disponibles
- Les schémas contenant le string dans leur nom soumis comme argument de **Schéma**
- Seuls les schémas installés
- Seuls les schémas qui peuvent être mis à niveau

Syntaxe

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

Si aucun argument de **Schéma** n'est soumis, alors toutes les schémas disponibles sont recensés. Autrement, les schémas sont recensés par des options soumises (*voir l'exemple ci-dessous*). Notez que vous pouvez soumettre l'argument de **Schéma** de nombreuses fois.

Options
La commande **list** accepte les options suivantes :

`--installed, --i`	Recenser uniquement la liste des schémas installés. Le réglage par défaut est **faux**.
`--upgradeable, --u`	Recenser uniquement les schémas lorsque des mises à niveau (patchs) sont disponibles. Le réglage par défaut est **faux**.
`--help, --h`	Afficher l'aide pour la commande.

Exemples

- Pour exécuter tous les schémas disponibles, exécutez : **xmlschemamanager list**
- Pour recenser les schémas installés, exécutez : **xmlschemamanager list -i**
- Pour recenser tous les schémas qui contiennent soit "doc", soit "nitf" dans leur nom, exécutez **xmlschemamanager list doc** :

4.1.8.5.6 reset

Cette commande supprime tous les schémas installés et le répertoire de mise sous cache. Vous réinitialiserez complètement votre environnement de schéma. Une fois avoir exécuté cette commande, vous devrez exécuter la commande `initialize`[140], pour pouvoir recréer le répertoire de mise sous cache. En alternative, exécuter la commande **reset** avec l'option **-i**. Pusique **reset -i** restaure l'installation originale du produit, nous vous recommandons que vous exécutiez la commande `update`[144] après avoir réalisé la réinitialisation et l'initialisation. En alternative, exécutez la commande **reset** avec les options **-i** and **-u**

Syntaxe

```
<exec> reset [options]
```

Options
La commande **reset** accepte les options suivantes :

`--init, --i`	Initialiser Gestionnaire de schéma après le reset. Le réglage par défaut est **faux**.
`--update, --u`	Mettre à jour la liste de schémas disponibles dans le cache. Le réglage par défaut est **faux**.
`--silent, --s`	Afficher uniquement des messages d'erreur. Le réglage par défaut est **faux**.
`--verbose, --v`	Afficher des informations détaillées lors de l'exécution. Le réglage par défaut est **faux**.
`--help, --h`	Afficher l'aide pour la commande.

Exemples

- Pour réinitialiser Gestionnaire de schéma, exécuter : **xmlschemamanager reset**
- Pour réinitialiser Gestionnaire de schéma et l'initialiser, exécutez : **xmlschemamanager reset -i**
- Pour réinitialiser Gestionnaire de schéma, initialiser-le et mettez à jour sa liste de schéma, exécutez : **xmlschemamanager reset -i-u**

4.1.8.5.7 uninstall

Cette commande désinstalle un ou plusieurs schémas. Par défaut, tout schéma référencé par la taxonomie actuelle sera également désinstallé. Pour désinstaller uniquement le schéma actuel et garder les schémas référencés, définir l'option `--k`.

Syntaxe

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

Pour désinstaller de multiples schémas, ajoutez l'argument **Schéma** de nombreuses fois.

L'argument **Schéma** est l'un des suivants :

- Un identifiant de schéma (avoir un format de **<name>-<version>**, par exemple : **cbcr-2.0**). Pour trouver les identifiants de schéma qui sont installés, exécutez la commande `list -i` 142 . Vous pouvez aussi utiliser un nom de schéma abrégé s'il est unique, par exemple **docbook**. Si vous utilisez un nom abrégé, alors tous les schémas qui contiennent une abréviation dans leur nom seront désinstallés.
- Le chemin vers un fichier **.altova_xmlschemas** téléchargé depuis le site web d'Altova. Pour information sur ces fichiers, voir *Introduction à SchemaManager : Comment cela fonctionne-t-il ?* 129.

Options
La commande **désinstaller** accepte les options suivantes :

`--keep-references, --k`	Définir cette option pour garder les schémas référencés. Le réglage par défaut est **faux**.
`--silent, --s`	Afficher uniquement des messages d'erreur. Le réglage par défaut est **faux**.
`--verbose, --v`	Afficher des informations détaillées lors de l'exécution. Le réglage par défaut est **faux**.
`--help, --h`	Afficher l'aide pour la commande.

Exemple

La commande suivante désinstalle les schémas CBCR 2.0 et EPUB 2.0 et leurs dépendances :

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

La commande suivante désinstalle la taxonomie **eba-2.10** mais pas les schémas qu'elle référence :

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

4.1.8.5.8 update

Cette commande requête la liste des schémas disponibles depuis l'emplacement de stockage en ligne et met à jour le répertoire de mise sous cache local. Vous devriez exécuter cette commande sauf si vous avez réalisé un `reset` 143 et `initialize` 140.

Syntaxe

```
<exec> update [options]
```

Options
La commande **mise à jour** accepte les options suivantes :

`--silent, --s`	Afficher uniquement des messages d'erreur. Le réglage par défaut est **faux**.
`--verbose, --v`	Afficher des informations détaillées lors de l'exécution. Le réglage par défaut est **faux**.

`--help, --h`	Afficher l'aide pour la commande.

Exemple

La commande suivante met à jour le cache local avec la liste des derniers schémas :

```
xmlschemamanager update
```

4.1.8.5.9 upgrade

Cette commande met à niveau toutes les schémas installés qui peuvent être mis à niveau à la dernière version *patchée* disponible. Vous pouvez identifier des schémas à mettre à niveau en exécutant la commande `list -u`[142].

Note : La commande **`mettre à niveau`** supprime une Police dépréciée si aucune version plus récente n'est disponible.

Syntaxe

```
<exec> upgrade [options]
```

Options

La commande **`mise à niveau`** accepte les options suivantes :

`--silent, --s`	Afficher uniquement des messages d'erreur. Le réglage par défaut est **`faux`**.
`--verbose, --v`	Afficher des informations détaillées lors de l'exécution. Le réglage par défaut est **`faux`**.
`--help, --h`	Afficher l'aide pour la commande.

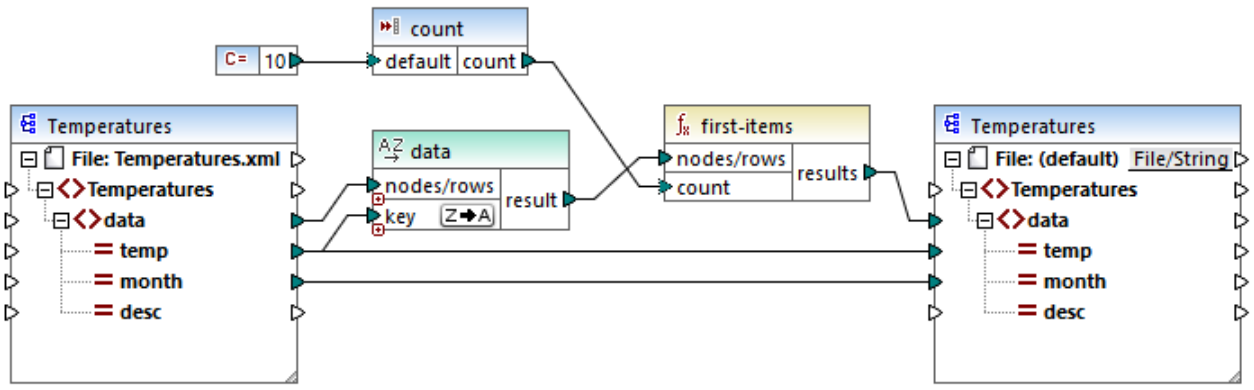
5 Composants de transformation

Cette section décrit les composants de transformation qui peuvent être utilisés pour transformer les données ou les stocker temporairement pour traitement ultérieur. La liste des composants de transformation est recensée ci-dessous :

Notez que les fonctions appartiennent également aux composants de transformation. Toutefois, les fonctions 194 sont organisées en section autonome.

5.1 Entrée simple

Si vous devez créer un mappage qui prend des paramètres en tant qu'entrée, vous pouvez le faire en ajoutant un type de composant spécial appelé "composant d'entrée simple". Les composants d'entrée simple possèdent toujours un type de données simple (par exemple, string, entier, etc.) au lieu d'une structure d'items et de séquences. Par exemple, dans le mappage illustré ci-dessous, il existe un composant d'entrée simple **count**. Son rôle est de fournir en tant que paramètre le nombre maximum de lignes qui doit être extrait depuis le fichier XML de source (avec la valeur **10** en tant que défaut). Chose importante, les nœuds fournis en tant qu'entrée de la fonction first-items[280] sont triés avec l'aide d'un composant sort, afin que le mappage sorte uniquement les températures *N* les plus élevées, où *N* est la valeur du paramètre.



FindHighestTemperatures.mfd

Une autre utilisation assez commune des composants d'entrée simple est de fournir un nom de fichier au mappage. Cela est utile dans des mappages qui lisent des fichiers d'entrée ou qui écrivent des fichiers de sortie dynamiquement, voir Traiter plusieurs fichiers d'entrée ou de sortie dynamiquement[400]. Dans le fichier XSLT généré, les composants d'entrée simple correspondent aux paramètres de feuille de style.

Vous pouvez créer chaque composant d'entrée simple (ou paramètre) comme optionnel ou obligatoire (voir Ajouter des composants d'entrée simple[148]). Le cas échéant, vous pouvez aussi créer des valeurs par défaut pour les paramètres d'entrée de mappage voir Créer une valeur d'entrée par défaut[150]. Ceci vous permet d'exécuter le mappage en toute sécurité, même si vous ne fournissez pas explicitement une valeur de mappage au moment de l'exécution. Pour consulter un exemple, voir Exemple : Utiliser des noms de fichier en tant que Paramètres de mappage[151].

Les paramètres d'entrée ajoutés dans la zone de mappage principale ne doivent pas être confondus avec les paramètres d'entrée dans les fonctions définies par l'utilisateur[203]. Il existe des similarités et des différences entre les deux comme suit.

Paramètres d'entrée sur le mappage	Paramètres d'entrée des fonctions définies par l'utilisateur
Ajouté depuis le menu **Fonction | Insérer entrée**.	Ajouté depuis le menu **Fonction | Insérer entrée**.
Peut avoir des types de données simples (string, entier, etc.).	Peut avoir des types de données simples et complexes.

Paramètres d'entrée sur le mappage	Paramètres d'entrée des fonctions définies par l'utilisateur
Applicable à tout le mappage.	Applicable uniquement dans le contexte de la fonction dans laquelle ils ont été définis.

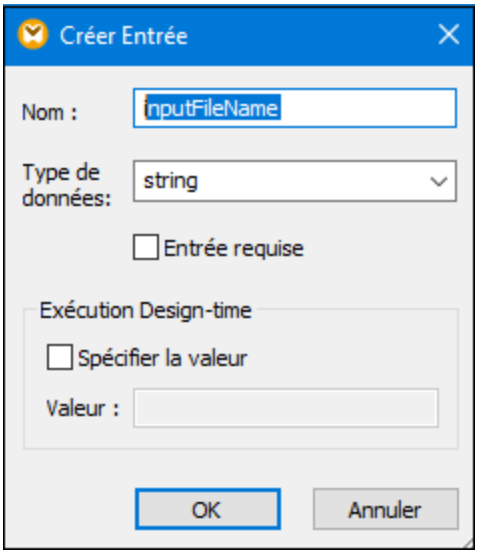
Lorsque vous avez créez un mappage inversé (utiliser la commande de menu **Outils | Créer le mappage inversé**), un composant d'entrée simple devient un composant de sortie simple.

5.1.1 Ajouter des composants d'entrée simple

Pour ajouter une entrée simple dans le mappage :

1. Veuillez vous assurer que la fenêtre de mappage affiche le mappage principal (pas une fonction définie par l'utilisateur).
2. Choisir une des options suivantes :

   - Dans le menu **Fonction**, cliquer sur **Insérer entrée**.
   - Dans le menu **Insertion**, cliquer sur **Insérer entrée**.
   - Cliquer sur la touche de la barre d'outils **Insérer entrée** .



3. Saisir un nom et choisir le type de données requis pour cette entrée. Si l'entrée doit être traitée comme un paramètre de mappage obligatoire, cocher la case **Entrée requise**. Pour une liste complète de paramètres, voir Paramètres de composant d'entrée simple 149.

Note : Le nom du paramètre peut contenir uniquement des lettres, des chiffres et des tirets bas ; aucun autre caractère n'est autorisé. Cela permet à un mappage de fonctionner sur tous les langages de génération de code.

4. Cliquer sur **OK**.

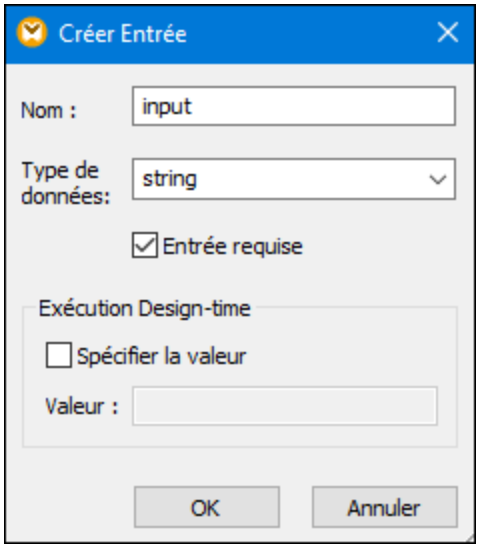
Vous pouvez changer ultérieurement tous les paramètres définis ici (voir Paramètres de composant d'entrée simple 149).

5.1.2 Paramètres de composant d'entrée simple

Vous pouvez définir les paramètres applicables à un composant d'entrée simple en l'ajoutant à la zone de mappage. Vous pouvez aussi changer les paramètres ultérieurement, depuis le dialogue Éditer entrée.

Pour ouvrir le dialogue Éditer entrée, choisir une des options suivantes :

- Sélectionner le composant et, dans le menu **Composant**, cliquer sur **Propriétés**.
- Double-cliquer sur le composant.
- Cliquer avec la touche de droite sur le composant puis cliquer sur **Propriétés**.



Dialogue Éditer entrée

Pour ouvrir le dialogue Éditer entrée, suivre une des étapes suivantes :

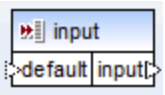
- Choisir le composant et, dans le menu **Composant**, cliquer sur **Propriétés**.
- Double-cliquer sur le composant.
- Cliquer avec la touche de droite sur le composant, et puis cliquer sur **Propriétés**.

Les paramètres disponibles sont les suivants.

Nom	Saisir un nom descriptif pour le paramètre d'entrée correspondant à ce composant. Au moment du temps d'exécution de mappage, la valeur saisie dans cette fenêtre de texte devient le nom du paramètre fourni au mappage ; ainsi aucune espace ni aucun caractère spécial n'est autorisé.
Type de données	Par défaut, tout paramètre d'entrée est traité en tant que type de données de string. Si le paramètre doit avoir un type de données différent, choisir la valeur respective depuis la liste. Lorsque le mappage est exécuté, MapForce envoie le paramètre d'entrée vers le type de donnée sélectionné ici.
Entrée est requise	Si activé, le paramètre rend le paramètre d'entrée obligatoire (c'est à dire, le mappage ne peut pas être exécuté à moins que vous fournissiez une valeur de paramètre). Désactiver cette case à cocher si vous spécifiez une valeur par défaut pour le paramètre d'entrée (voir Créer une valeur d'entrée par défaut 150).
Spécifier valeur	Ce paramètre est applicable uniquement si vous exécutez le mappage pendant la période de conception du design, en cliquant sur l'onglet **Aperçu**. Il vous permet de saisir directement dans le composant la valeur à utiliser en tant que l'entrée de mappage.
Valeur	Ce paramètre est uniquement applicable si vous exécutez le mappage pendant la période de conception du design, en cliquant sur sur l'onglet **Aperçu**. Pour saisir une valeur à utiliser par MapForce en tant qu'entrée de mappage, sélectionner la case à cocher Spécifier valeur, puis saisir la valeur requise. **Note :** Si vous cochez la case **Spécifier valeur** et saisissez une valeur dans la case adjacente, la valeur saisie prend précédence sur la valeur par défaut lorsque vous consultez le mappage (c'est à dire au moment de l'exécution au moment du design). Néanmoins, la valeur au moment du design n'a pas d'effet dans le XSLT, XQuery, ou de code de programme généré, dans exécution par MapForce Server, ou le déploiement sur FlowForce Server.

5.1.3 Créer une valeur d'entrée par défaut

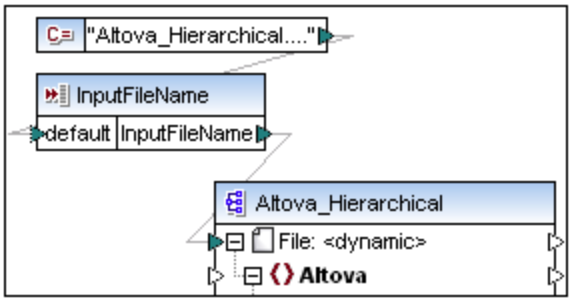
Une fois avoir ajouté un composant d'entrée dans la zone de mappage, veuillez noter l'item de **défaut** situé à gauche du composant.



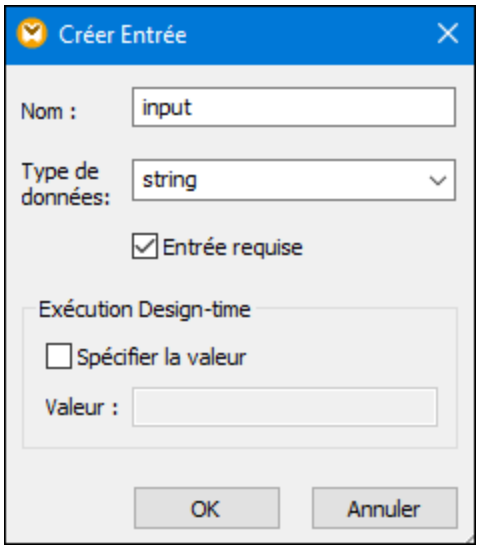
Composant d'entrée simple

L'item par défaut vous permet de vous connecter à une valeur par défaut optionnelle à ce composant d'entrée, comme suit :

1. Ajouter un composant de constante (dans le menu **Insérer**, cliquer sur **Constant**), puis le connecter dans l'item **default** du composant d'entrée.



2. Double-cliquez le composant d'entrée et désactivez la case à cocher **Entrée requise**. Lorsque vous créez une valeur d'entrée par défaut, ce paramètre n'est pas significatif et entraîne des avertissements de validation de mappage.



3. Cliquer sur **OK**.

Note : Si vous cochez la case **Spécifier valeur** et saisissez une valeur dans la case adjacente, la valeur saisie prend précédence sur la valeur par défaut lorsque vous consultez le mappage (c'est à dire au moment de l'exécution au moment du design). Néanmoins, la valeur au moment du design n'a pas d'effet dans le XSLT, XQuery, ou de code de programme généré, dans exécution par MapForce Server, ou le déploiement sur FlowForce Server.

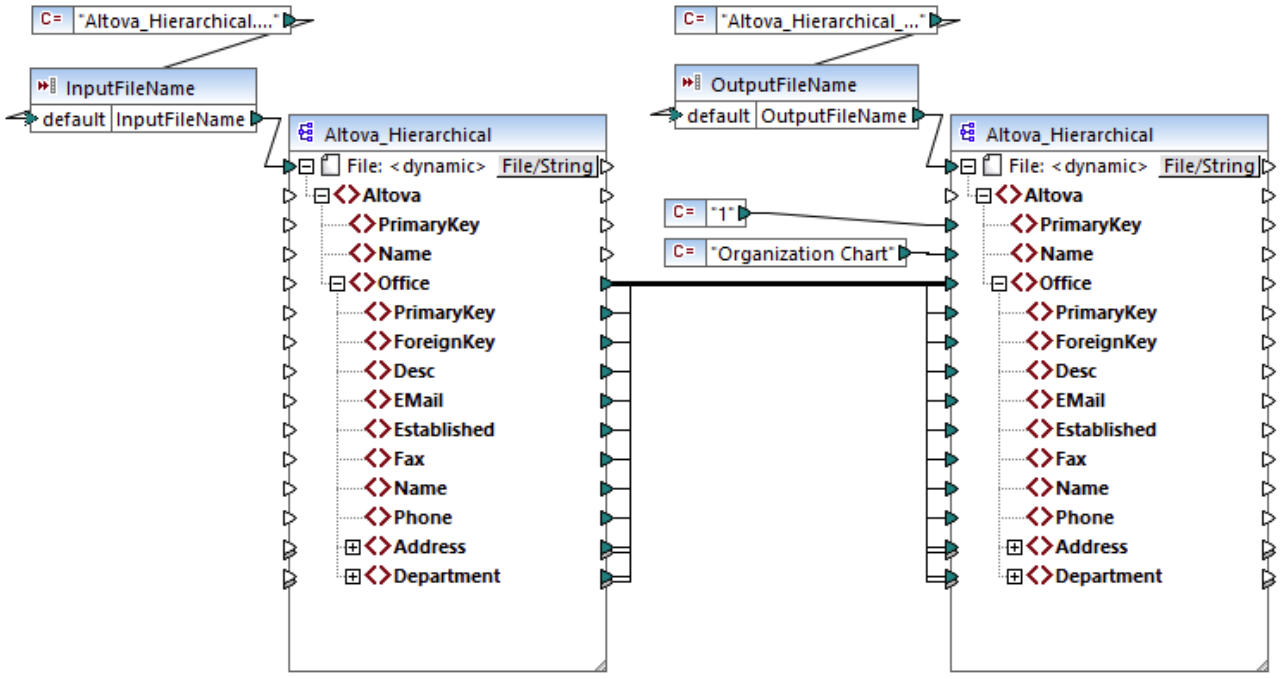
5.1.4 Exemple : Utiliser les noms de fichier en tant que paramètres de mappage

Cet exemple vous guide dans les étapes nécessaires pour exécuter un mappage qui prend les paramètres d'entrée lors du runtime. Le fichier de conception de mappage utilisé dans cet exemple est disponible dans le chemin suivant :
<Documents>\Altova\MapForce2024\MapForceExamples\FileNamesAsParameters.mfd.

Ce mappage lit des données depuis un fichier XML de source et l'écrit dans un fichier XML cible. Les données sont écrites dans le fichier cible de manière presque inchangée ; seuls les attributs **PrimaryKey** sont **Name**

remplis avec des valeurs constantes depuis le mappage. L'objectif principal du mappage est de permettre à l'appelant de spécifier le nom du fichier d'entrée, en tant que paramètres de mappage, au moment du runtime du mappage.

Pour ce faire, le mappage a deux composants d'entrée : **InputFileName** et **OutputFileName**. Ceux-ci apportent le nom d'entrée du fichier (et le nom du fichier de sortie respectivement) du fichier XML de source et de cible. Pour cette raison, ils sont connectés à l'item **Fichier : <dynamic>**. Vous pouvez passer un composant dans ce mode en cliquant sur la touche **File** (File) , et en sélectionnant **Utiliser Noms de fichier dynamique fournis par le mappage**.



FileNamesAsParameters.mfd (MapForce Enterprise Edition)

Si vous double-cliquez sur la barre de titre du composant **InputFileName** ou **OutputFileName**, vous pouvez consulter ou éditer leurs propriétés. Par exemple, vous pouvez spécifier le type de données du paramètres d'entrée ou changer le nom de paramètre d'entrée, comme décrit dans Paramètres de composant d'entrée simple 149. Dans cet exemple, les paramètres d'entrée et de sortie sont configurés comme suit :

- Le paramètre **InputFileName** est de type "string" et il a une valeur par défaut fourni par une constante définie dans le même mappage. La constante est de type "string" et sa valeur est "Altova_Hierarchical.xml". Ainsi, lorsque ce mappage est exécuté, il tentera de lire des données depuis un fichier nommé "Altova_Hierarchical.xml", en partant du principe que vous ne fournissez pas d'autres valeurs en tant que paramètre.
- Le paramètre **OutputFileName** est de type "string" et il a aussi une valeur par défaut fournie par une constante définie dans le même mappage. La constante est de type "string" et sa valeur est "Altova_Hierarchical_output.xml". C'est pourquoi le mappage créera un fichier de sortie XML appelé "Altova_Hierarchical_output.xml" lorsqu'il est exécuté, en partant du principe que vous ne fournissez pas d'autres valeurs en tant que paramètres.

Les sections suivantes illustrent comment exécuter le mappage et fournir des paramètres dans les langages de transformation suivants :

- XSLT 2.0 153, utiliser RaptorXML Server

XSLT 2.0

Si vous générez du code dans XSLT 1.0, XSLT 2.0 ou XSLT 3.0, un fichier batch **DoTransform.bat** est généré dans le répertoire de cible choisi, en plus du fichier XSLT. Le fichier **DoTransform.bat** vous permet d'exécuter le mappage avec RaptorXML Server, voir Automatisation avec RaptorXML Server 426.

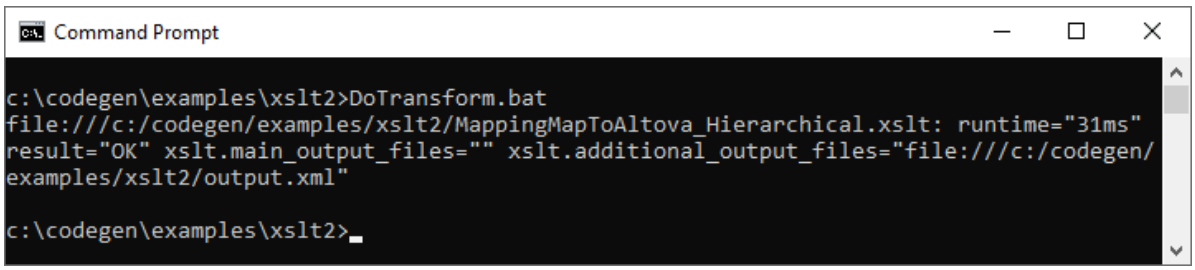
Pour utiliser un fichier d'entrée (ou de sortie) différent, éditer le fichier **DoTransform.bat** pour inclure les paramètres requis, comme suit :

1. Tout d'abord, générer le code XSLT. Par exemple, pour générer XSLT 2.0, sélectionner la commande de menu **Fichier | Générer du Code dans | XSLT 2.0**.
2. Copier le fichier **Altova_Hierarchical.xml** depuis **<Documents>\Altova\MapForce2024\MapForceExamples** dans le répertoire où vous avez généré le code XSLT 2.0 (dans cet exemple, **c:\codegen\examples\xslt2**). Comme indiqué précédemment, le mappage tentera de lire ce fichier si vous ne fournissez pas de valeur personnalisée dans le paramètre **InputFileName**.
3. Éditer **DoTransform.bat** pour inclure le paramètre d'entrée personnalisé soit avant soit après **%***. Veuillez noter que la valeur de paramètre est contenue entre des guillemets simples. Les paramètres d'entrée disponibles sont recensés dans la section **rem** (Remarque). Supposons que vous souhaitez générer un fichier de sortie appelé **output.xml**. Pour ce faire, changer le fichier **DoTransform.bat** comme suit :

```
@echo off

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

Lorsque vous exécutez le fichier DoTransform.bat, RaptorXML Server termine la transformation en utilisant **Altova_Hierarchical.xml** en tant que paramètres d'entrée. Si vous avez suivi les étapes ci-dessus, le nom du fichier de sortie généré sera **output.xml**.



5.2 Sortie simple

Un composant de sortie (ou "sortie simple") est un composant MapForce qui vous permet de retourner une valeur string provenant du mappage. Les composants de sortie représentent uniquement un type possible type de composants de cible[30], mais ne doivent pas être confondus avec ce dernier. Utiliser un composant de sortie simple lorsque vous souhaitez retourner une valeur de string depuis le mappage. Dans la zone de mappage, les composants de sortie simple jouent un rôle simple d'un composant cible qui a un type de données string à la place d'une structure d'items et de séquences. Par conséquent, vous pouvez créer un composant de sortie simple au lieu (ou en plus de) d'un composant sur base de fichier. Par exemple, vous pouvez utiliser un composant de sortie simple pour tester et consulter rapidement la sortie d'une fonction (voir Exemple : Consulter la sortie de fonction[156]).

Les composants de sortie simple ne devraient pas être confondus avec les paramètres de sortie de fonctions définies par l'utilisateur (voir Fonctions définies par l'utilisateur[203]). Il existe des similarités et des différences entre les deux comme suit.

Composants de sortie	Paramètres de sortie des fonctions définies par l'utilisateur
Ajouté depuis le menu **Fonction | Insérer sortie**.	Ajouté depuis le menu **Fonction | Insérer sortie**.
A "string" en tant que type de données.	Peut avoir des types de données simples et complexes.
Applicable à tout le mappage.	Applicable uniquement dans le contexte de la fonction dans laquelle ils ont été définis.

Le cas échéant, vous pouvez ajouter plusieurs composants de sortie simples à un mappage. Vous pouvez aussi utiliser des composants de sortie simples en combinaison avec des composants cibles basés sur fichier. Si votre mappage contient plusieurs composants cibles, vous pouvez consulter les données retournées par un composant particulier en cliquant sur la touche **Aperçu** ( ) dans la barre de titre du composant, puis en cliquant sur l'onglet **Sortie** dans la fenêtre Mappage.

Vous pouvez utiliser les composants de sortie simples comme suit dans les langages de transformation MapForce :

Langage	Comment cela fonctionne ?
XSLT 1.0, XSLT 2.0, XSLT 3.0	Dans les fichiers XSLT générés, un composant de sortie simple défini dans le mappage devient la sortie de la transformation XSLT. Si vous utilisez RaptorXML Server, vous pouvez instruire RaptorXML Server pour écrire la sortie de mappage sur le fichier transmis en tant que valeur dans le paramètre `--output`. Pour écrire la sortie dans un fichier, ajouter ou éditer dans le paramètre `--output` dans le fichier **DoTransform.bat**. Par exemple, le fichier **DoTransform.bat** suivant a été édité pour écrire la sortie de mappage dans le fichier **Output.txt** (voir texte marqué).

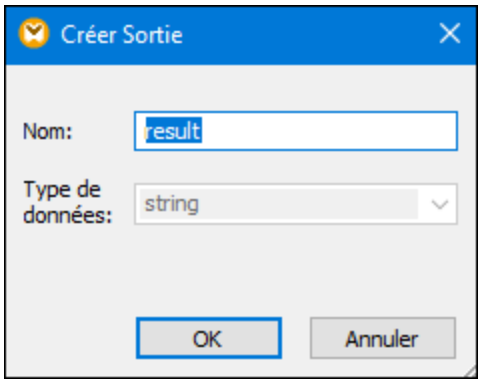
Langage	Comment cela fonctionne ?
	`RaptorXML xslt --xslt-version=2 --input="MappingMapToResult1.xslt" --output="Output.txt" %* "MappingMapToResult1.xslt"` Si un paramètre `--output` n'est pas défini, la sortie de mappage sera écrite dans le flux de sortie standard (stdout) lorsque le mappage est exécuté.

Si vous créez un mappage inversé (en utilisant les commandes de menu **Outils | Créer mappage inverse**), le composant de sortie simple devient un composant d'entrée simple.

5.2.1 Ajouter des composants de sortie simples

Pour ajouter un composant de sortie vers la zone de mappage :

1. Veuillez vous assurer que la fenêtre de mappage affiche le mappage principal (n'est pas une fonction définie par l'utilisateur).
2. Choisir une des étapes suivantes :
   a. Dans le menu **Fonction**, cliquer sur **Insérer sortie**.
   b. Cliquer sur la touche de la barre d'outils **Insérer sortie**  .
3. Saisir un nom pour le composant.
4. Cliquer sur **OK**.



Dialogue Créer sortie

Vous pouvez changer le nom du composant à tout moment ultérieurement, d'une des manières suivantes :

- Sélectionner le composant et, dans le menu **Composant**, cliquer sur **Propriétés**.
- Double-cliquer sur l'en-tête de composant.

- Cliquer avec la touche de droite sur l'en-tête de composant, puis cliquer sur **Propriétés**.

5.2.2 Exemple : Consulter la sortie de fonction

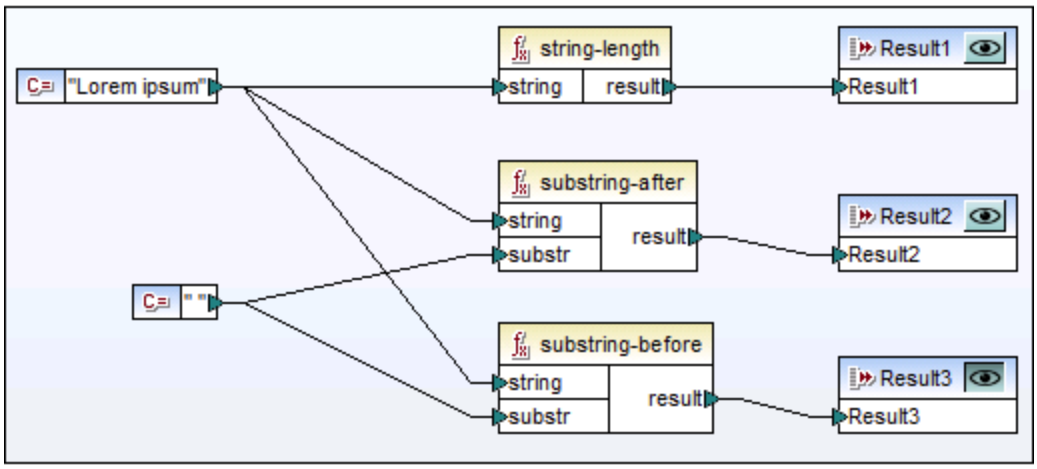
Cet exemple illustre comment consulter la sortie retournée par les fonctions MapForce à l'aide de composants de sortie simples. Vous profiterez au maximum de cet exemple si vous possédez des connaissances de base des fonctions en général et des fonctions MapForce en particulier. Si vous ne connaissez pas les fonctions de MapForce, vous pouvez consulter Utiliser les fonctions 194 avant de continuer.

Notre objectif est d'ajouter un nombre de fonctions dans la zone de mappage et d'apprendre comment consulter leur sortie à l'aide de composants de sortie simples. En particulier, l'exemple utilise quelques fonctions simples disponibles dans la bibliothèque centrale. Voici un sommaire de leur utilisation :

`string-length` 310	Retourne le nombre de caractères contenus dans le string fourni en tant qu'argument. Par exemple, si vous transmettez à cette fonction la valeur "Lorem ipsum", le résultat est "11", étant donné qu'il s'agit du nombre de caractères contenus dans le texte "Lorem ipsum".
`substring-after` 312	Retourne la partie du string qui se produit après le séparateur fourni en tant qu'argument. Par exemple, si vous passez dans cette fonction la valeur "Lorem ipsum" et le caractère d'espace (" "), le résultat sera "ipsum".
`substring-before` 312	Retourne la partie du string qui se produit avant le séparateur fourni en tant qu'argument. Par exemple, si vous passez dans cette fonction la valeur "Lorem ipsum" et le caractère d'espace (" "), le résultat sera "Lorem".

Pour tester chacune de ces fonctions par rapport à une valeur de texte personnalisée ("Lorem ipsum", dans cet exemple), suivre les étapes ci-dessous :

1. Ajoutez une constante avec la valeur "Lorem ipsum" à la zone de mappage (utilisez la commande de menu **Insérer | Constante**). La constante sera le paramètre d'entrée pour chacune des fonctions à tester.
2. Ajoutez les fonctions **`string-length`**, **`substring-after`** et **`substring-before`** à la zone de mappage, en les glissant vers la zone de mappage depuis la bibliothèque core, la section **string functions**.
3. Ajouter une constante avec une espace vide (" ") en tant que valeur. Cela sera le paramètre de séparateur requis par les fonctions **`substring-after`** et **`substring-before`**.
4. Ajoutez trois composants de sortie simple (utiliser la commande de menu **Fonction | Insérer sortie**). Dans cet exemple, ils ont été nommés *Result1*, *Result2* et *Result3*, bien que vous puissiez leur donner un autre titre.
5. Connecter les composants comme indiqué ci-dessous.



Tester la sortie de fonction avec des composants de sortie simples

Comme indiqué dans l'échantillon ci-dessus, le string "Lorem ipsum" agit en tant que paramètre d'entrée pour chaque fonction **`string-length`**, **`substring-after`** et **`substring-before`**. De plus, les fonctions **`substring-after`** et **`substring-before`** prennent une valeur d'espace dans leur deuxième paramètre d'entrée. Les composants **Result1**, **Result2** et **Result3** peuvent être utilisés pour consulter le résultat de chaque fonction.

Pour consulter la sortie de toute fonction :

- Cliquez sur la touche **Aperçu** () dans la barre de titre du composant, puis cliquez sur l'onglet **Sortie** dans la fenêtre de mappage.

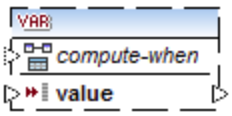
5.3 Les variables

Une variable est un type de composant spécial utilisé pour stocker un résultat de mappage intermédiaire pour un traitement ultérieur. Les variables peuvent être de type simple (par exemple, string, entier, booléennes, etc) ou de type complexe (une structure arborescente). Voir les exemples des deux types dans les sous-rubriques ci-dessous.

Un des aspects les plus importantes concernant les variables est qu'elles sont des séquences, et qu'elles peuvent être utilisées pour créer des séquences. Le terme *séquence* signifie une liste d'items zéro ou plus. Cela permet à une variable de traiter plusieurs items pour la durée du cycle de vie du mappage. Pour plus d'information, voir aussi Règles et stratégies de mappage 405. Toutefois, il est également possible d'assigner la valeur à une variable une fois et préserver cette valeur pour le reste du mappage. Pour des détails, voir Changer le contexte et l'étendue des variables 164.

Variables simples

Une simple variable est créée pour représenter des types atomiques tels que les strings, nombres booléennes (*voir la capture d'écran ci-dessous*).



Variables complexes

Une variable complexe a une structure d'arborescence. Les structures sur lesquelles une variable complexe peut être basée sont résumées dans la liste ci-dessous.

Édition de base de MapForce :

- Structures de schéma XML

Édition professionnelle de MapForce :

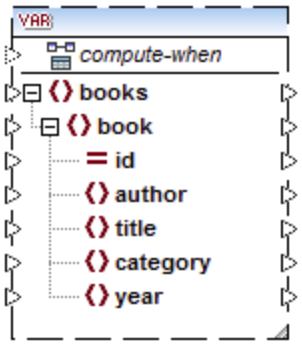
- Structure de schéma XML
- Structure de base de données

Édition MapForce Enterprise :

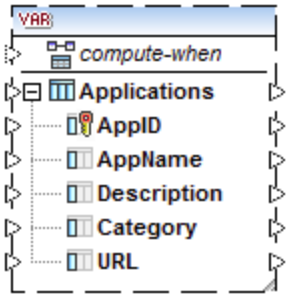
- Structure de schéma XML
- Structure de base de données
- Structure EDI
- Structure FlexText
- Structure de schéma JSON

Exemple 1 : Variable basée sur Schéma XML

Vous pouvez créer une variable de type complexe en fournissant un schéma XML qui définit la structure de la variable (*voir la capture d'écran ci-dessous*). Si le schéma définit des éléments globalement, vous pouvez choisir lequel doit devenir le nœud de racine de la structure variable. Veuillez noter qu'une variable n'a pas de fichier XML d'instance associé. Les données de la variable sont calculées durant l'exécution du mappage.



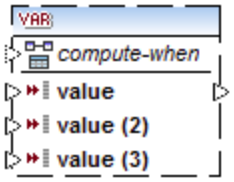
Exemple 2 : Variable basée sur la base de données (éditions MapForce Professional et Enterprise)
Si vous choisissez une structure de base de données pour votre variable(*voir la capture d'écran ci-dessous*), vous pouvez choisir une table de base de données spécifique en tant qu'item de racine pour la structure de variable. MapForce vous permet de créer des variable basés sur BD avec une arborescence de tables associées. L'arborescence de tables associées représente une structure in-memory qui n'a pas de connexion vers la base de données lors de l'exécution. Ceci signifie également qu'il n'y a pas de gestion automatique de la clé étrangère et pas d'actions de table dans les paramètres ou variables.



Compute-when
Dans les deux exemples ci-dessus, chaque variable est dotée d'un élément dénommé `compute-when`. Connecter cet élément est optionnel : Ceci vous permet de contrôler comment la valeur de variable doit être calculée dans le mappage. Pour plus d'informations, voir Changer le contexte et l'étendue des variables 164.

Variables avec des entrées dupliquées

Si nécessaire, les items d'une structure de variable peuvent être dupliqués pour accepter les données de plus d'une connexion source. Ceci est semblable aux entrées dupliquées 40 dans les composants standard. Néanmoins, cela ne s'applique pas aux variables créées depuis les tables de base de données. La capture d'écran ci-dessous illustre une variable simple avec des entrées dupliquées.



Mappages en chaîne par rapport aux variables

Les variables peuvent être comparées aux composants intermédiaires d'un mappage en chaîne[94]. Néanmoins, les variables sont plus flexibles et pratiques si vous ne devez pas produire de fichiers intermédiaires à chaque étape du mappage. La table suivante souligne les différences entre les variables et les mappages en chaîne.

Mappages enchaînés	Variables
Les mappages en chaîne impliquent deux étapes indépendantes. Par exemple, un mappage a trois composants, notamment A, B et C. Étape 1 : mapper des données de A à B. Étape 2 : mapper des données de B à C.	Vous pouvez contrôler quand et combien de fois la valeur variable est calculée quand le mappage est exécuté. Pour les détails, voir Changer le contexte et l'étendue des variables[164].
Lorsque le mappage est exécuté, les résultats intermédiaires seront stockés extérieurement dans des fichiers.	Lorsque le mappage est exécuté, des résultats intermédiaires sont stockés intérieurement. Aucun fichier externe contenant des résultats d'une variable n'est produit.
Le résultat intermédiaire peut être consulté en utilisant la touche d'aperçu.	Un résultat de variable ne peut pas être consulté, puisqu'il est calculé lors de l'exécution du mappage.

Note : les variables ne sont pas prises en charge si le langage de transformation de mappage est réglé sur XSLT 1.0.

5.3.1 Ajouter des variables

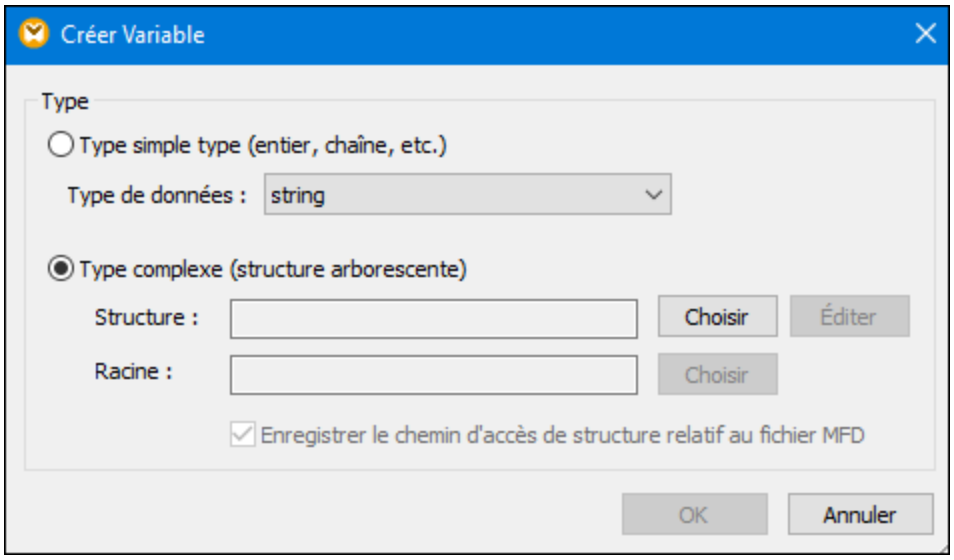
Cette rubrique explique comment ajouter une variable au mappage. La première option est d'ajouter une variable par la commande de menu ou de la barre d'outils. La deuxième option vous permet d'ajouter une variable par le menu contextuel.

Option 1 : par la commande de menu ou de la barre d'outils

Cette option vous permet d'ajouter une variable par la commande de menu ou une barre d'outils. Suivez les étapes ci-dessous :

1. Allez au menu **Insérer** and cliquez sur **Variable**. En alternative, cliquez sur 

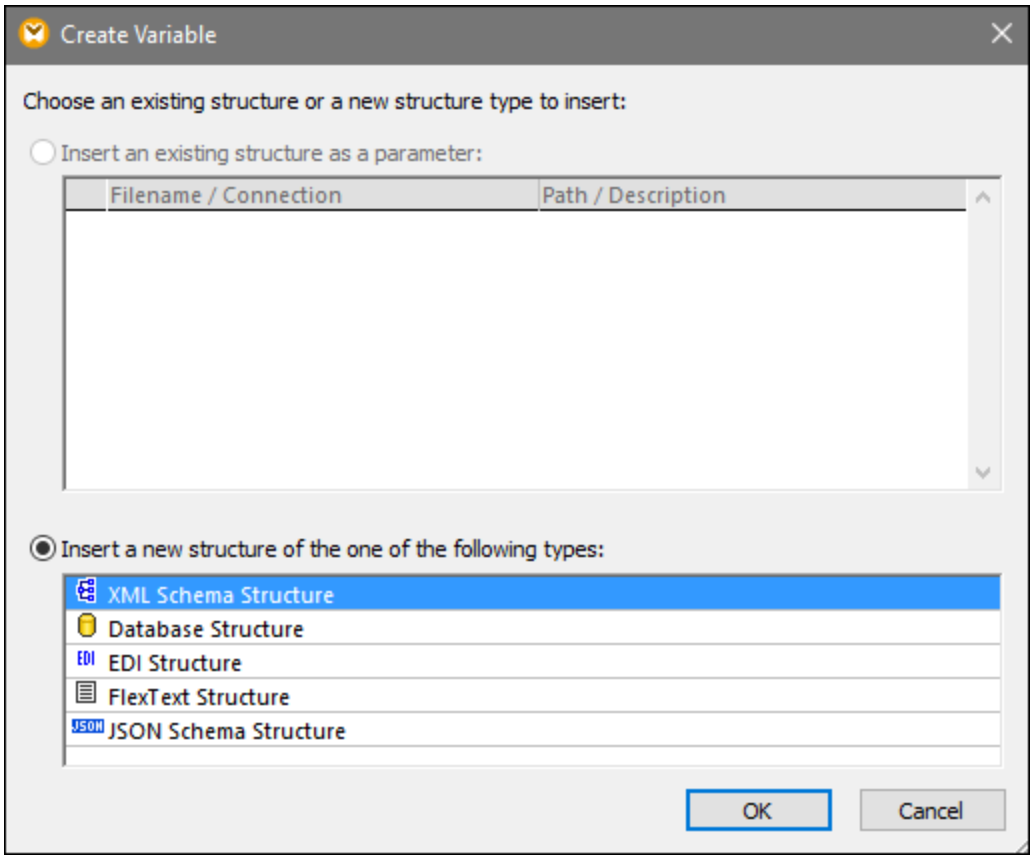
bouton de la barre d'outils (**Variable**).



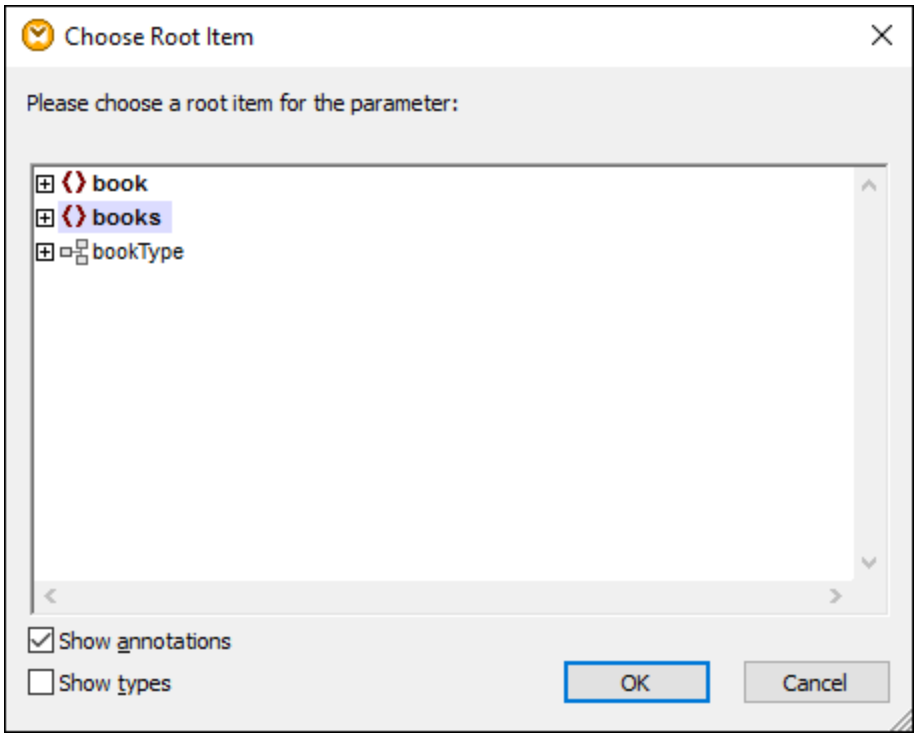
2. Sélectionnez le type de variable que vous souhaitez insérer (type simple ou complexe).

Si vous sélectionnez **type complexe**, vous devez suivre quelques étapes supplémentaires :

3. Cliquez sur **Choisir** pour sélectionner la source qui devrait fournir la structure de la variable[158]. Les structures illustrées dans la capture d'écran s'appliquent uniquement à MapForce Enterprise Edition. Voir la liste de structures pertinentes à d'autres éditions de MapForce dans la rubrique précédente[158] .



4. Lorsque vous serez invité à le faire, spécifiez un item de racine de la structure de variable. Par exemple, dans les schémas XML, vous pouvez sélectionner tout élément ou type de la source sélectionnée (*voir la capture d'écran ci-dessous*).

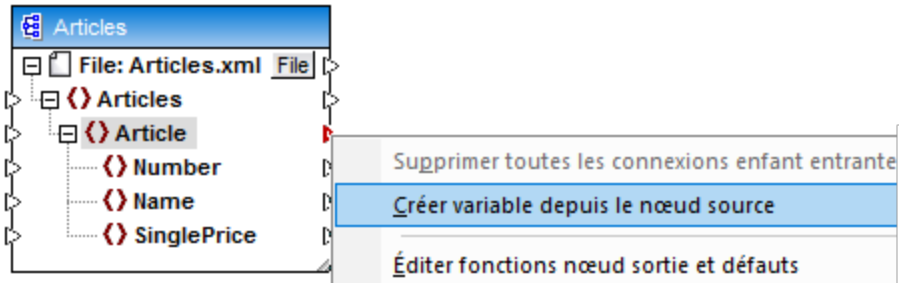


Option 2 : par le menu contextuel

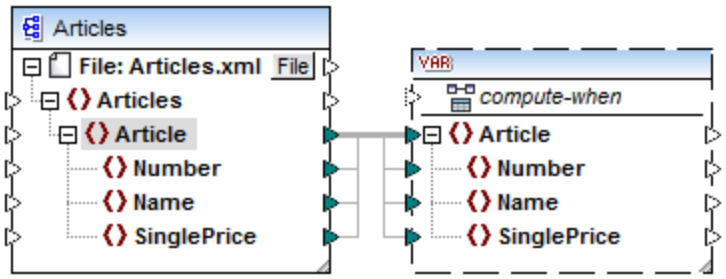
La deuxième option vous permet de créer une variable utilisant le menu contextuel. Les options possibles sont recensées ci-dessous.

Variable depuis un nœud source

Pour créer une variable depuis un nœud source, cliquez de la touche droite sur le connecteur de sortie d'un composant (dans cet exemple, le connecteur de sortie de l'élément `<Article>`) et sélectionnez **Créer une variable depuis le nœud source** (*voir la capture d'écran ci-dessous*).



Ceci crée une variable complexe avec le schéma source du composant `Articles`. Tous les éléments sont automatiquement connectés avec une connexion copy-all [55] (*voir la capture d'écran ci-dessous*).

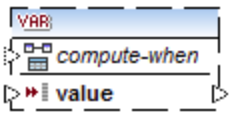


Variable depuis un nœud cible
Pour créer une variable depuis un nœud cible, cliquez de la touche droite sur le connecteur d'entrée d'un composant cible et sélectionnez **Créer une variable depuis un nœud cible**. Ceci crée une variable complexe avec le même schéma que pour la cible. Tous les items sont automatiquement connectés avec une connexion copy-all.

Variable depuis un filtre :
Pour créer une variable utilisant un filtre, cliquez de la touche droite sur le connecteur de sortie d'un composant de filtre (`on-true/on-false`) et sélectionnez **Créer une variable depuis le nœud source**. Cela crée un composant complexe avec le schéma de source, et utilise automatiquement l'item lié à l'entrée de filtre en tant que l'élément racine du composant intermédiaire.

5.3.2 Changer le contexte et l'étendue des variables

Chaque variable a un élément d'entrée `compute-when` (*voir la capture d'écran ci-dessous*), qui vous permet de contrôler l'étendue de la variable. Ceci signifie que vous pouvez contrôler quand et combien de fois la valeur de variable est calculée quand le mappage est exécuté. Vous ne devez pas connecter cette entrée dans de nombreux cas, mais il peut être essentiel d'écraser le contexte par défaut ou d'optimiser la performance de mappage.



Les termes suivants sont pertinents pour la discussion sur l'étendue et le contexte des variables : *subtree* et *variable value*. Une sous-arborescence est un ensemble d'items/nœuds dans un composant cible et tous ses descendants : par exemple, un élément `<Person>` avec ses éléments enfant `<FirstName>` et `<LastName>`.

Une valeur variable signifie les données qui sont disponibles du côté sortie du composant variable.

- Pour des variables simples, il s'agit d'une séquence de valeurs atomiques dont le type de données est spécifié dans les propriétés de composant.
- Pour les variables complexes, il s'agit d'une séquence de nœuds de racine (du type spécifié dans les propriétés de composant), chacune incluant tous ses nœuds descendants.

La séquence des valeurs atomiques (ou nœuds) peut contenir un ou plusieurs éléments zéro. Cela dépend de ce qui est connecté du côté entrée de la variable, et des items de parents dans les composants source et cible.

« Compute-when » n'est pas connecté (défaut)

Si l'item d'entrée `compute-when` n'est pas connecté vers un nœud de sortie d'un composant de source, la valeur de variable est calculée *à chaque fois qu'il est utilisé la première fois dans une sous-arborescence de cible* soit directement par le biais d'un connecteur depuis le composant de variable vers un nœud dans le composant de cible, ou indirectement par le biais des fonctions. La même valeur de variable est aussi utilisée pour tous les nœuds enfants de cible dans la sous-arborescence.

La valeur de variable réelle dépend des connexions entre les items de parent des composants de source et de cible. Ce comportement par défaut est le même que celui des sorties complexes des fonctions régulières définies par l'utilisateur 214 et les appels de fonction de service Web. Si la sortie de variable est connectée à plusieurs nœuds de cible non-liés, la valeur de variable est calculée *séparément pour chacun d'entre eux*. Cela peut produire des résultats différents dans chaque cas, parce que plusieurs connexions de parent influent sur le contexte dans lequel la valeur de la variable est évaluée.

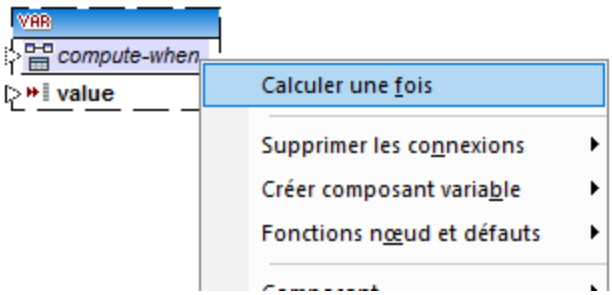
« Compute-when » est connecté

En connectant un connecteur de sortie d'un composant source vers `compute-when`, la variable est calculée *à chaque fois que cet item source est utilisé la première fois dans une sous-arborescence cible*.

La variable agit comme si elle était un item enfant de l'item connecté à `compute-when`. Cela permet de lier la variable à un item de source spécifique. Concrètement, lors du runtime, la variable est réévaluée lorsqu'un nouvel item est lu depuis la séquence dans le composant de source. Cela est lié aux règles générales qui gouvernent les connexions dans MapForce : Pour chaque item source, un item cible est créé. Dans ce cas, `compute-when` donne l'instruction à MapForce de calculer la valeur variable pour chaque item source. Pour plus d'information, voir Règles et stratégies de mappage 405.

« Compute-once »

Si nécessaire, vous pouvez choisir de calculer la valeur de variable *une fois avant chaque composants de cible*, rendant la variable essentiellement une constante globale pour le reste du mappage. Pour ce faire, cliquer avec la touche de droite sur l'item `compute-when` et sélectionner **Calculer une fois** depuis le menu contextuel :

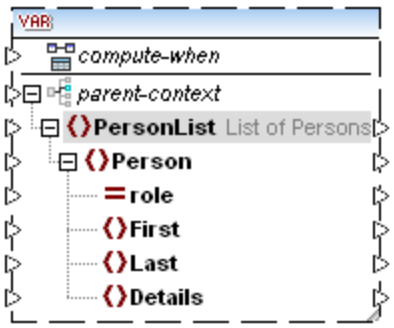


Lorsque vous changez l'étendue d'une variable vers `compute-when=once`, le connecteur d'entrée est supprimé de l'item `compute-when`, puisqu'une telle variable est uniquement évaluée une fois. Dans une fonction définie

par l'utilisateur, la variable `compute-when=once` est évaluée à chaque fois que la fonction est appelée avant que le résultat de fonction réel soit évalué.

Parent-context

Ajouter un item `parent-context` peut être nécessaire, par exemple si votre mappage utilise plusieurs filtres et que vous nécessitez un nœud de parent supplémentaire pour les itérations. Pour plus de détails, voir Exemple : 412. Changer le contexte Parent 412. Pour ajouter un parent-context, cliquez avec la touche de droite sur le nœud racine (dans cet exemple, `PersonList`) et sélectionnez **Ajouter parent- context** depuis le menu contextuel. Cela ajoute un nouveau nœud, `parent-context`, à la hiérarchie existante.

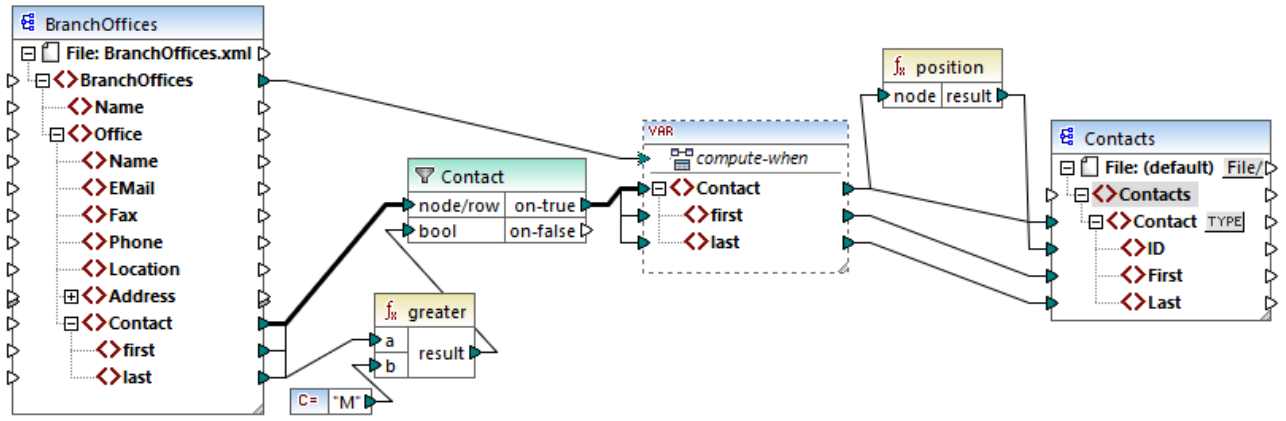


Le contexte parent ajoute un nœud parent virtuel à la hiérarchie dans le composant. Cela vous permet d'itérer sur un nœud supplémentaire dans le même composant de source ou dans un composant de source différent.

5.3.3 Exemple : Filtre et numéroter les nœuds

Le mappage illustré dans cet exemple est disponible sous **PositionInFilteredSequence.mfd** dans le dossier **<Documents>\Altova\MapForce2024\MapForceExamples**.

Ce mappage lit un fichier XML qui contient les coordonnées de plusieurs personnes, les filtre et les écrit dans un fichier XML cible. L'objectif du mappage est de filtrer depuis le fichier XML de source uniquement les personnes dont le nom de famille commence avec la lettre "M" ou une des lettres suivantes de l'alphabet. Ensuite, les contacts extraits doivent être numérotés. Le nombre va servir d'identifiant unique de chaque contact dans le fichier XML cible.



PositionInFilteredSequence.mfd

Pour atteindre l'objectif ci-dessus, les types de composant suivants ont été ajoutés au mappage :

- Un filtre (voir Filtres et conditions 176)
- Une variable complexe (voir Ajouter des variables 160)
- Les fonctions **greater** 260 et **position** 297 (voir Ajouter une fonction au mappage 195)
- Une constante (Pour ajouter une constante, sélectionner la commande de menu **Insérer | Constante**).

La variable utilise le même schéma que le composant de source. Si vous cliquez de la touche droite sur la variable et sélectionnez **Propriétés** depuis le menu contextuel, notez que le nœud **BranchOffices/Office/Contact** est sélectionné comme nœud racine pour cette structure de variable.

Tout d'abord, les données du composant de source sont passées dans le filtre. Le filtre passe ensuite à la variable les enregistrements qui remplissent la condition de filtre. Concrètement, le filtre est configuré ou obtenir uniquement les nœuds `Contact` où le prénom est égal ou plus grand que "M". Pour ce faire, la fonction **greater** 260 compare chaque item `last` avec la valeur de constante "M".

La variable a l'entrée `compute-when` connectée à l'item de racine du composant de source (`BranchOffices`). Lors du runtime, cela entraîne une réévaluation de la variable à chaque fois qu'un nouvel item est lu depuis la séquence dans le composant de source. Dans ce mappage, néanmoins, la connexion ou la non-connexion de l'item `compute-when` ne fait aucune différence. La raison est que la variable est connectée à l'item de source `Contact` (indirectement par le filtre), et il calculera autant de fois qu'il y a d'instances de `Contact` qui se conforment à la condition de filtre.

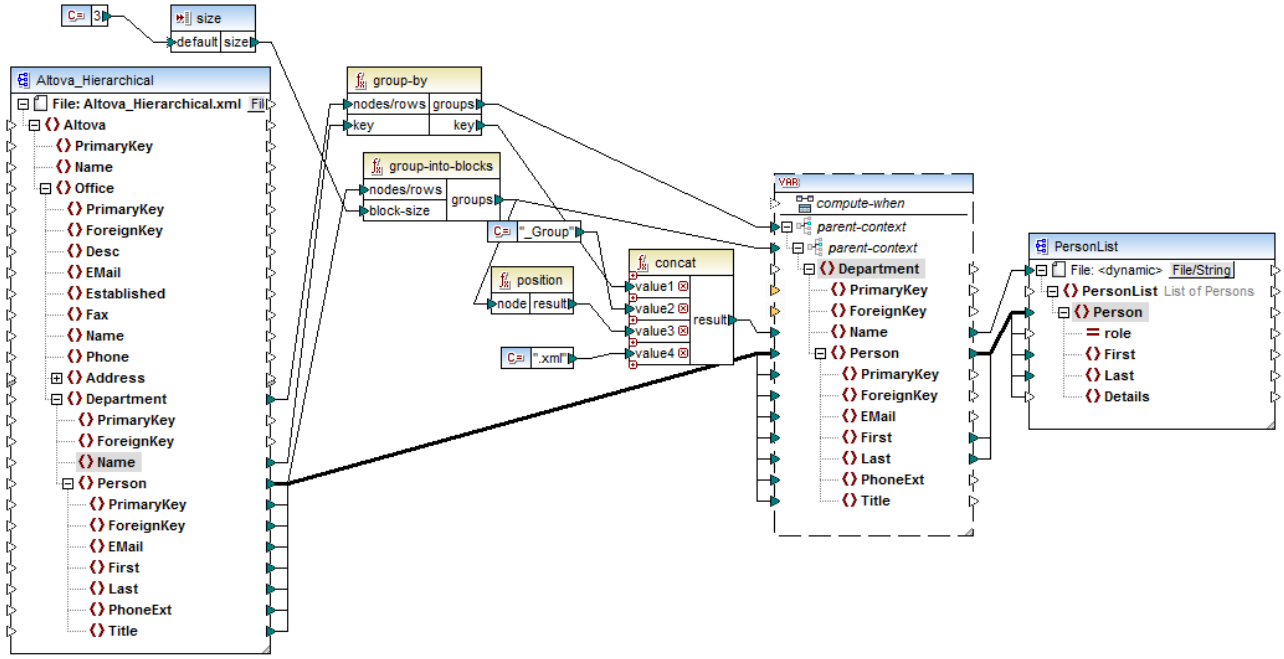
Les fonctions **position** 297 retournent, pour chaque itération de la variable, le nombre de séquences actuelles. Seuls huit contacts remplissent la condition de filtre ; c'est pourquoi, si vous consultez le mappage et regardez la sortie, il est recommandé de noter comment les ID 1 à 8 ont été écrites dans l'élément `ID` du composant de cible.

La variable a été nécessaire, à cause de l'exigence de numéroter tous les enregistrements. Si nous avions connecté le résultat de filtre directement au composant de cible, il n'y aurait pas moyen de numéroter chaque occurrence de `Contact`. L'objectif de la variable dans ce mappage est donc de stocker chaque instance de `Contact` temporairement dans le mappage, elle peut donc être numérotée avant d'être écrite dans la cible.

5.3.4 Exemple : Grouper et sous-grouper des enregistrements

Le mappage illustré dans cet exemple est disponible sous **DividePersonsByDepartmentIntoGroups.mfd** dans le dossier **<Documents>\Altova\MapForce2024\MapForceExamples**.

Ce mappage traite un fichier XML qui contient des enregistrements d'employés d'une compagnie fictionnelle. L'entreprise a deux bureaux : "Nanonull, Inc." et "Nanonull Partners, Inc". Chaque bureau possède plusieurs départements (par exemple, "IT", "Marketing", etc.), et chaque département compte un ou plusieurs employés. L'objectif du mappage est de créer des groupes d'un maximum de trois personnes de chaque département, quelque soit le bureau. La taille de chaque groupe est de trois par défaut ; néanmoins, cela devrait être simple à changer le cas échéant. Chaque groupe doit être enregistré en tant que fichier XML séparé, et le nom doit présenter le format "<Department Name>_GroupN" (par exemple, **Marketing_Group1.xml**, **Marketing_Group2.xml**, etc.).



DividePersonsByDepartmentIntoGroups.mfd

Comme illustré ci-dessus, afin de pouvoir atteindre l'objectif de mappage, une variable complexe a été ajoutée au mappage, et quelques autres types de composant (surtout des fonctions). La variable présente la même structure qu'un item `Department` dans le XML de source. Si vous cliquez avec la touche de droite sur la variable pour consulter ses propriétés, vous remarquerez qu'elle utilise le même schéma XML que le composant de source, et contient `Department` en tant qu'élément de racine. Chose importante, la variable comporte deux items `parent-context` imbriqués, qui assurent que la variable est calculée tout d'abord dans le contexte de chaque département, et puis dans le contexte de chaque groupe dans le cadre de chaque département (voir aussi Changer le contexte et l'étendue des variables 164).

Au début, le mappage itère à travers tous les départements pour obtenir le nom de chaque département (cela sera ensuite exigé pour créer le nom de fichier correspondant à chaque groupe). Cela est réalisé en connectant la fonction **`group-by`** 284 dans l'item de source `Department`, et en fournissant le nom du département en tant que clé de regroupement.

Ensuite, dans le contexte de chaque département, un deuxième groupement a lieu : le mappage appelle la fonction **group-into-blocks** 290 pour créer les groupes des personnes requis. La taille de chaque groupe est fournie par un composant d'entrée simple d'une valeur par défaut de "3". La valeur par défaut est fournie par une constante. Dans cet exemple, afin de changer la taille de chaque groupe, il est facilement possible de modifier la valeurs constante selon vos besoins. Néanmoins, le composant d'entrée "size" peut aussi être modifié de manière à ce que, si le mappage est exécuté par le code généré ou avec MapForce Server, la taille de chaque groupe peut être fournie en tant qu'un paramètre dans le mappage. Pour plus d'informations, voir Fournir des paramètres au mappage 147.

Enfin, la valeur de la variable est fournie dans le composant XML PersonList cible. Le nom de fichier pour chaque groupe créé a été calculé en concaténant les parties suivantes, par le biais de la fonction **concat** 307 :

1. Le nom de chaque département
2. Le string "_Group"
3. Le numéro du groupe dans la séquence actuelle (par exemple, "1" s'il s'agit du premier groupe pour ce département)
4. Le string ".xml"

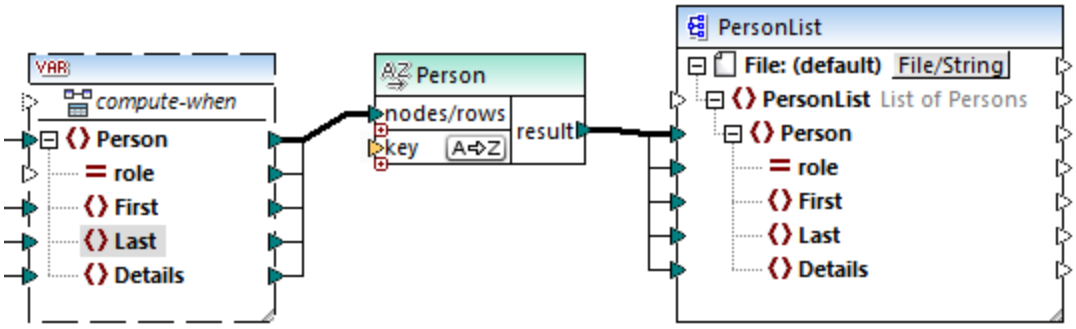
Le résultat de cette concaténation est stocké dans l'item `Name` de la variable, puis est fourni en tant que nom de fichier dynamiques dans le composant cible. Cela entraîne la création d'un nouveau nom de fichier pour chaque valeur reçue. Dans cet exemple, la variable calcule huit groupes au total, donc huit fichiers de sortie sont créés lorsque le mappage est exécuté, le cas échéant. Pour plus d'informations concernant cette technique, voir Traiter plusieurs fichiers d'entrée ou de sortie dynamiquement 400.

5.4 Trier les données

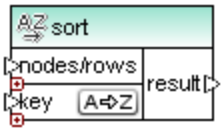
Afin de trier les données basées sur une clé de tri spécifique, utilisez un composant Sort (Tri). Le composant Tri prend en charge les langues cible suivantes : XSLT2, XQuery, et Built-in.

Afin d'ajouter un composant de tri au mappage, suivez une des étapes suivantes :

- Cliquez avec la touche de droite sur une connexion existante et sélectionnez **Insérer Sort : Nœuds/Lignes** depuis le menu contextuel. Cela permet d'insérer le composant Tri et de le connecter automatiquement aux composants de source et de cible. Par exemple, dans le mappage ci-dessous, le composant Tri a été inséré entre une variable et un composant XML. La seule chose qui reste à connecter manuellement est la clé de triage (le champ avec lequel vous souhaitez trier).



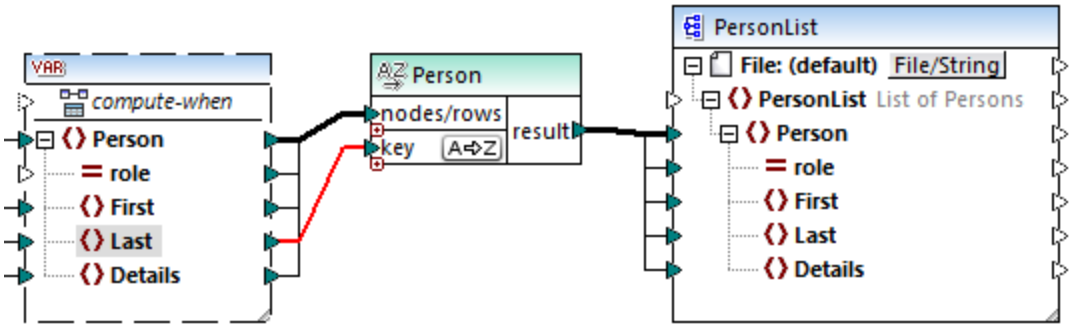
- Dans le menu **Insérer**, cliquer sur **Sort** (ou bien cliquer sur la touche de la barre d'outils **Sort** ). Le composant Sort est inséré dans sa forme "non-connectée".



  Dès qu'une connexion a été établie dans le composant de source, le nom de barre de titre change pour prendre celui de l'item connecté à l'item `nodes/rows`.

Pour définir l'item par lequel vous souhaitez trier :

- Connecter l'item par lequel vous souhaitez trier le paramètre `key` du composant Tri. Par exemple, dans le mappage ci-dessous, les nœuds/lignes `Person` sont triées par le champ `Last`.

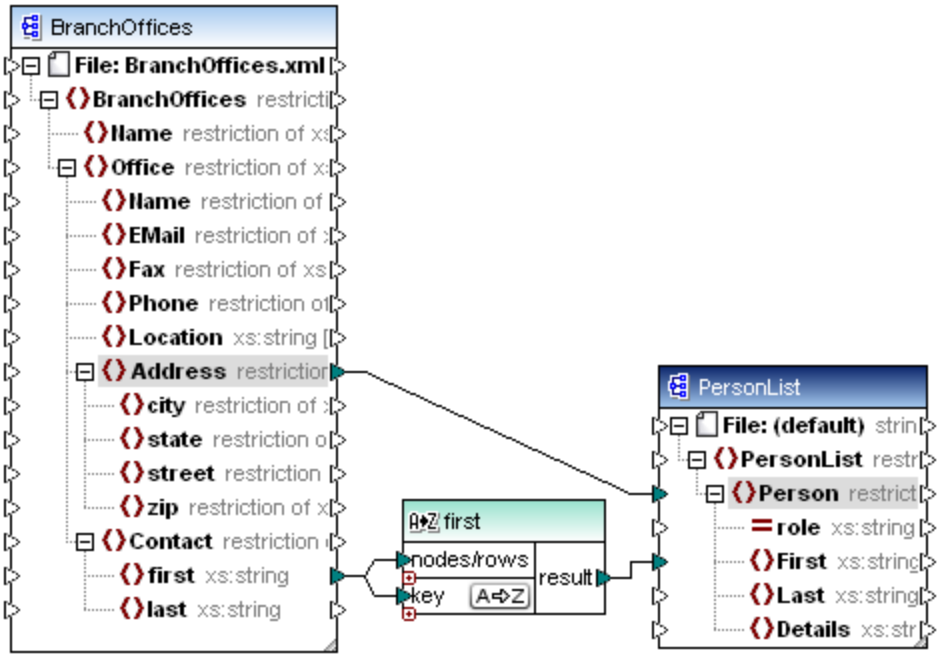


Pour modifier l'ordre de tri :

- Cliquer sur l'icône A⇨Z dans le composant Sort. Elle passe à Z➔A pour montrer que l'ordre de tri a été inversé.

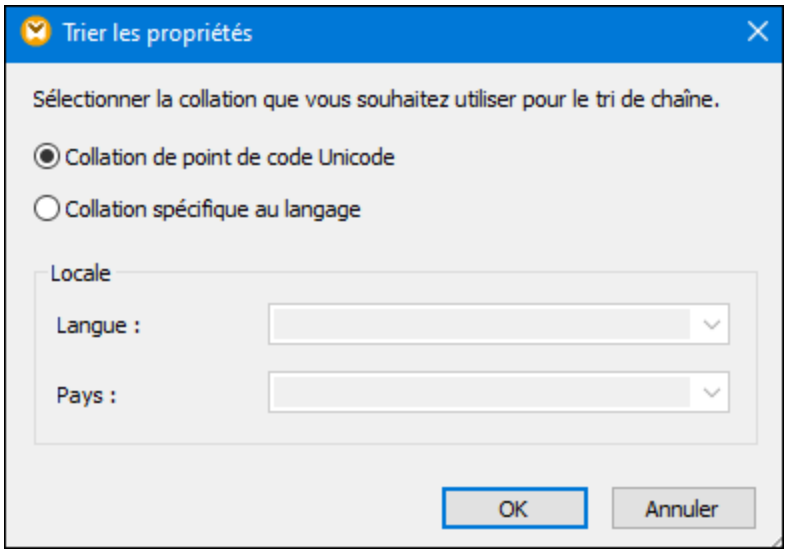
Pour trier des données d'entrée consistant en des items de type simple :

- Connecter l'item aux deux paramètres `nœuds/lignes` et `clé` du composant de tri sort. Dans le mappage ci-dessous, l'élément de type simple `first` est en cours de tri.



Pour trier les strings en utilisant des règles spécifiques au langage :

- Double-cliquer sur l'en-tête du composant Sort pour ouvrir le dialogue Trier les propriétés.



Collation point de code Unicode : Cette option (par défaut) compare/trie des strings basés sur des valeurs de point de code. Les valeurs de point de code sont des entiers qui ont été attribués à des caractères abstraits dans le Universal Character Set adopté par l'Unicode Consortium. Cette option permet un tri pour de nombreuses langues et scripts.

Collation spécifique à la langue : Cette option vous permet de définir la langue et les variantes régionales spécifiques par lesquelles vous souhaitez trier. Cette option est prise en charge lors de l'utilisation du moteur d'exécution BUILT-IN. Pour XSLT, la prise en charge dépend du moteur spécifique utilisé pour exécuter le code.

5.4.1 Trier par clés multiples

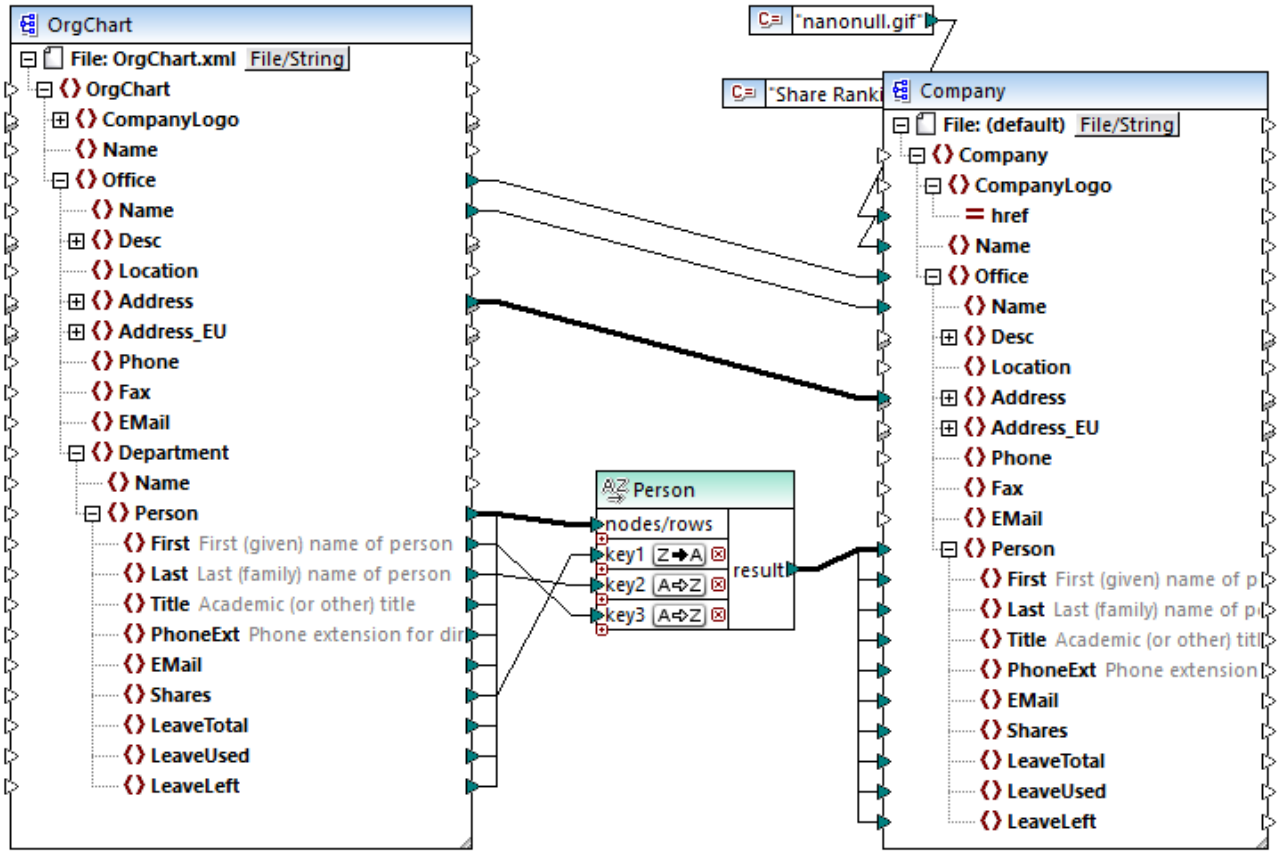
Une fois avoir ajouté un composant Sort au mappage, une clé de tri appelée `key` sera créée par défaut.

Composant Sort par défaut

Si vous souhaitez trier par clés multiples, ajuster le composant Sort comme suit :

- Cliquer sur l'icône **Ajouter clé** (⊡) pour ajouter une nouvelle clé (par exemple, `key2` dans le mappage ci-dessous).
- Cliquer sur l'icône **Delete clé** (⊠) pour supprimer une clé.
- Déposer une connexion dans l'icône ⊡ pour ajouter une clé et pour s'y connecter.

Un mappage qui illustre le tri avec plusieurs clés est disponible dans le chemin suivant : **<Documents>\Altova\MapForce2024\MapForceExamples\SortByMultipleKeys.mfd**.



SortByMultipleKeys.mfd

Dans le mappage ci-dessus, les enregistrements `Person` sont triés avec trois clés de tri :

1. `Shares` (nombre de parts que détient une personne)
2. `Last` (nom de famille)
3. `First` (prénom)

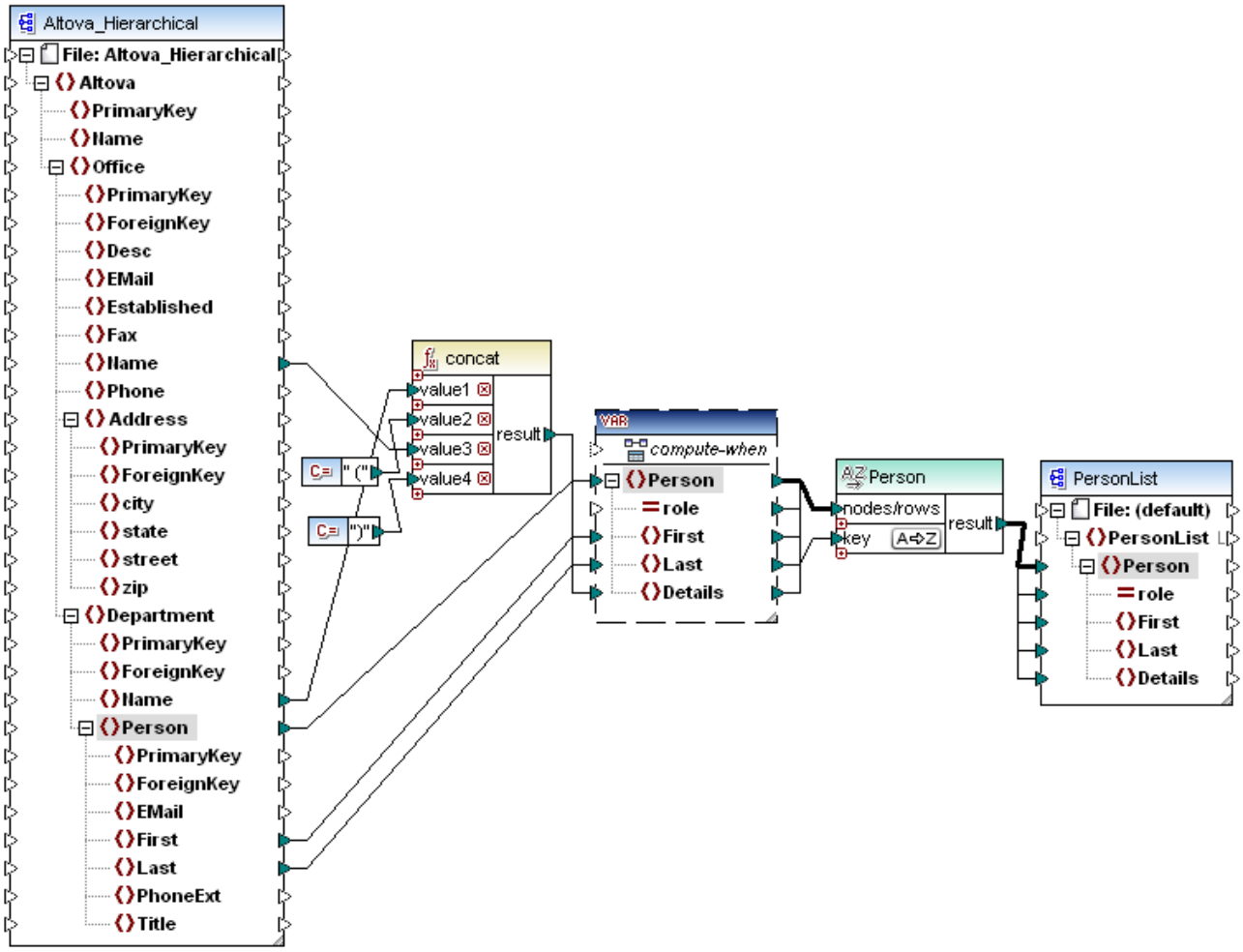
Veuillez noter que la position de la clé de tri dans le composant Sort détermine sa priorité de tri. Par exemple, dans le mappage ci-dessus, les enregistrements sont tout d'abord triés par le nombre de parts. Il s'agit de la clé de tri avec la priorité la plus élevée. Si le nombre de parts est le même, les gens sont ensuite triés par leur nom de famille. Enfin, si plusieurs personnes ont le même nombre de parts et le même nom de famille, le prénom de la personne sera pris en compte.

L'ordre de tri de chaque clé peut être différent. Dans le mappage ci-dessus, la clé `Shares` a un ordre de tri descendant (Z-A), alors que les deux autres clés ont un ordre de tri ascendant (A-Z).

5.4.2 Trier par variables

Dans certains cas, il peut s'avérer nécessaire d'ajouter des variables intermédiaires au mappage pour aboutir au résultat escompté. Cet exemple illustre comment extraire des enregistrements d'un fichier XML et les trier, avec l'aide des variables intermédiaires. L'exemple est accompagné par un échantillon de mappage situé dans

le chemin suivant :
<Documents>\Altova\MapForce2024\MapForceExamples\Altova_Hierarchical_Sort.mfd.



Altova_Hierarchical_Sort.mfd

Ce mappage lit des données provenant d'un fichier XML de source appelé **Altova_Hierarchical.xml** et l'écrit dans un fichier XML cible. Comme affiché ci-dessus, le XML source contient l'information à propos d'une entreprise fictive. L'entreprise est divisée en bureaux. Les bureaux sont sous-divisés en départements, et les départements sont répartis en personnes.

Le composant XML cible, `PersonList`, contient une liste des enregistrements `Person`. L'item `Details` est censé stocker des informations concernant le bureau et le département auquel la personne appartient.

L'objectif est d'extraire toutes les personnes depuis le XML de source et de les trier par ordre alphabétique avec leur nom de famille. De plus, le nom du bureau et de département auquel chaque personne appartient doit être écrit dans l'item `Details`.

Pour aboutir cet objectif, cet exemple utilise les types de composant suivants :

1. La fonction **`concat`**. Dans ce mappage, cette fonction retourne un string dans le format `Office(Department)`. Il prend en tant qu'entrée le nom du bureau, le nom du département et deux

constantes qui donnent les crochets de début et de fin. Voir aussi Ajouter une fonction au mappage[195] .

2. Une variable intermédiaire. Le rôle de la variable est d'apporter toutes les données concernant une personne dans le même contexte de mappage. La variable a l'effet suivant : le mappage consulte le département et le bureau de chaque personne dans le contexte de chaque personne. Autrement dit, la variable "se souvient" du nom du bureau et du département auquel appartient une personne. Sans la variable, le contexte serait incorrect, et le mappage produirait une sortie non désirée (pour plus d'informations concernant comment un mappage est exécuté, voir Mapper les règles et les stratégies[405]). Veuillez noter que la variable copie la structure du fichier XML cible (elle utilise le même schéma XML). Cela permet de connecter le résultat de tri vers la cible, par le biais d'une connexion Copier tout. Voir aussi Utiliser des variables[158] et Connexions Copier tout[55].
3. Un composant Sort, qui effectue le tri lui-même. Veuillez noter que l'entrée de clé du composant `Sort` est connecté à l'item `Last` de la variable, qui trie tous les enregistrements de personnes par leur nom de famille.

5.5 Filtres et conditions

Si vous souhaitez filtrer des données ou obtenir une valeur par condition, vous pouvez utiliser un des types de composant suivants :

- Filtre : Nœuds/Lignes ( )
- Condition If-Else ( )

Vous pouvez ajouter ces composants dans le mappage soit depuis le menu **Insérer**, soit depuis la barre d'outils **Insérer composant**. Attention, chaque composant indiqué ci-dessus possède son propre comportement et ses propres exigences. Les différences sont expliquées dans les sections suivantes.

Filtrer les nœud ou les lignes

Si vous souhaitez filtrer des données, y compris des nœuds XML , utiliser un composant **Filtrer nœuds/lignes**. Celui-ci vous permet d'extraire un sous-ensemble de nœuds depuis un ensemble de données plus important, sur la base d'une condition vrai ou faux. Sa structure dans la zone de mappage le reflète :

Dans la structure ci-dessus, la condition connectée à **bool** détermine si le **nœud/ligne** connecté va bien vers la sortie **on-true** ou **on-false**. Concrètement, si la condition est vraie, le **nœud/ligne** sera redirigé vers la sortie **on-true**. Ainsi, si la condition est fausse, le **nœud/ligne** sera redirigé vers la sortie **on-false**.

Si votre mappage nécessite de consommer uniquement des items qui *correspondent* à la condition de filtre, vous pouvez laisser la sortie **on-false** sans connexion. Si vous devez traiter les items qui *ne correspondent pas* à la condition de filtre, connecter la sortie **on-false** vers une cible dans laquelle ces items doivent être redirigés.

Pour un exemple de mappage étape par étape, voir Exemple : Filtrer des nœuds 177.

Retourner une valeur par condition

Si vous souhaitez obtenir une seule valeur (pas un nœud ou une ligne) de manière conditionnelle, utiliser une **Condition If-Else**. Veuillez noter que celles-ci ne sont pas appropriées pour filtrer les nœuds ou les lignes. Contrairement aux composants **Filtrer nœuds/lignes**, une **Condition If-Else** retourne une valeur de type simple (comme un string ou un entier). C'est pourquoi, les **Conditions If-Else** ne se prêtent que pour des scénarios dans lesquels vous devez traiter une valeur simple par condition. Par exemple, partons du principe que vous avez une liste de températures moyennes par mois, dans le format :

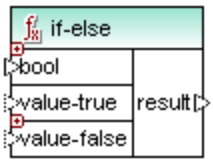
```
<Temperatures>
   <data temp="19.2" month="2010-06" />
   <data temp="22.3" month="2010-07" />
   <data temp="19.5" month="2010-08" />
   <data temp="14.2" month="2010-09" />
   <data temp="7.8" month="2010-10" />
   <data temp="6.9" month="2010-11" />
```



```
   <data temp="-1.0" month="2010-12" />
</Temperatures>
```

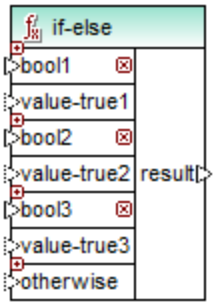
Une **Condition If-Else** vous permettrait de retourner, pour chaque item dans la liste, la valeur "high" si la température dépasse 20 degrés Celsius, et la valeur "low" si la température est inférieure à 5 degrés Celsius.

Dans le mappage, la structure de la **Condition If-Else** ressemble à l'exemple suivant :



Si la condition connectée à **bool** est vraie, alors la valeur connectée à **value-true** est sorti sous la forme de **result**. Si la condition est fausse, la valeur connectée à **value-false** est sortie sous la forme de **result**. Le type de donnée de **result** n'est pas connu à l'avance ; il dépend du type de données de la valeur connectée à **value-true** ou **value-false**. Attention : ce doit toujours être un type simple (string, entier, etc). La connexion de valeurs d'entrée de type complexe (comme des nœuds ou des lignes) n'est pas prise en charge par la **Condition If-Else**.

Les **Conditions If-Else** sont extensibles. Cela signifie que vous pouvez ajouter plusieurs conditions dans le composant en cliquant sur la touche **Ajouter** ( ). Pour supprimer une condition préalablement ajoutée, cliquer sur la touche **Supprimer** ( ). Cette fonction vous permet de contrôler plusieurs conditions et de retourner une valeur différente pour chaque condition, si elle est vraie.



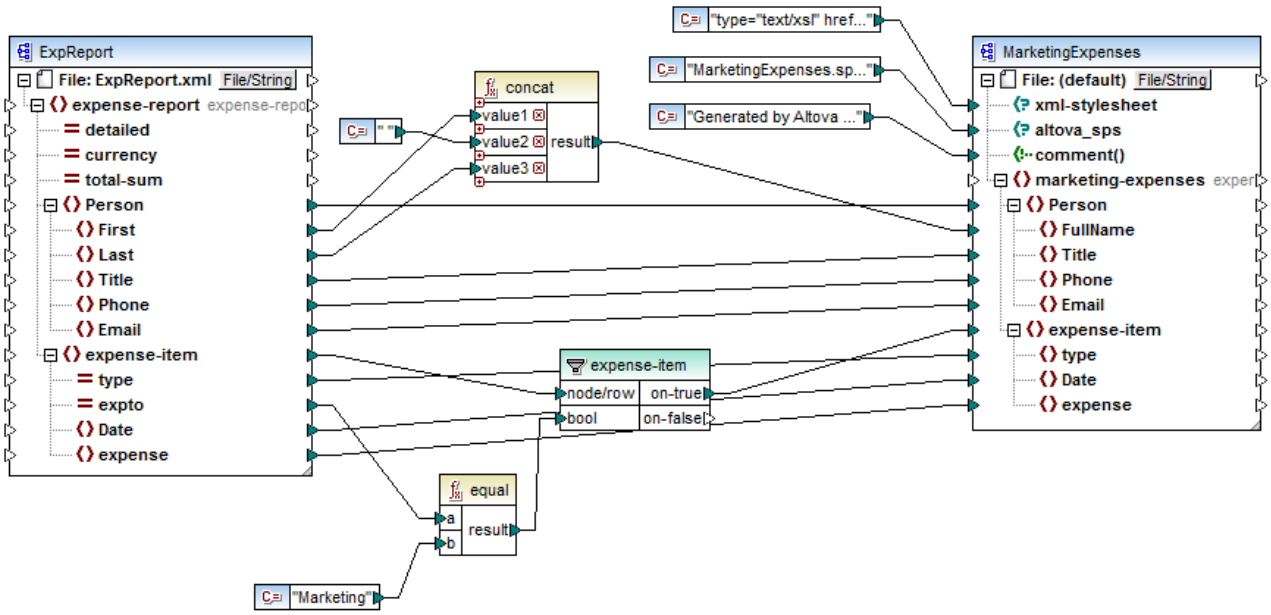
Les **Conditions If-Else** étendues sont évaluées du haut en bas (la première condition est contrôlée en premier, puis la deuxième, etc.). Si vous souhaitez retourner une valeur si aucune des conditions ne sont vraies, connectez-vous à **otherwise**.

Pour un exemple de mappage étape par étape, voir Exemple: Retourner une valeur par condition 179.

5.5.1 Exemple : Filtrer des nœuds

Cet exemple vous montre comment filtrer des nœuds basés sur une condition true/false. Un composant **Filtre : Nœuds/Lignes** ( ) est utilisé pour atteindre cet objectif.

Le mappage décrit dans cet exemple est disponible sous le chemin suivant :
<Documents>\Altova\MapForce2024\MapForceExamples\MarketingExpenses.mfd.



Comme indiqué ci-dessus, le mappage lit des données depuis un XML de source qui contient les notes de frais ("ExpReport") et écrit les données dans un XML cible ("MarketingExpenses"). Il existe plusieurs autres composants entre la cible et la source. Le composant le plus pertinent est le filtre **expense-item** ( ), qui représente le sujet de cette rubrique.

Le but du mappage est de filtrer uniquement les postes de dépense qui appartiennent au département Marketing. Pour atteindre cet objectif, un composant de filtre a été ajouté au mappage. (Pour ajouter un filtre, cliquez dans le menu **Insérer**, puis cliquez sur **Filtre : Nœuds/Lignes**.)

Afin d'identifier si chaque poste de dépense doit être attribué à Marketing, ce mappage consulte la valeur de l'attribut "expto" dans la source. Cet attribut a la valeur "Marketing" dès que la dépense est une dépense de marketing. Par exemple, dans la liste de code ci-dessous, le premier et le troisième poste de dépense correspond à Marketing, le deuxième appartient à Development, et le quatrième appartient à Sales :

```
...
   <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>
...
```

Entrée XML avant exécution du mappage

Dans la zone de mappage, l'entrée **nœud/ligne** du filtre est connectée au nœud **expense-item** dans le composant de source. Cela assure que le composant de filtre obtient la liste des nœuds qu'il doit traiter.

Pour ajouter la condition sur la base de laquelle le filtrage doit se produire, nous avons ajouté la fonction **equal** depuis la bibliothèque principale de MapForce (pour plus d'informations, voir aussi Ajouter une fonction au mappage 195). La fonction **equal** compare la valeur de l'attribut `expto` à une constante qui a la valeur `Marketing`. (Pour ajouter une constante, cliquez sur le menu **Insérer**, puis cliquez sur **Constante**.)

Puisque nous souhaitons filtrer uniquement les items qui satisfont à la condition, nous connectons uniquement la sortie **on-true** du filtre sur le composant de cible.

Lorsque vous consultez le résultat de mappage, en cliquant sur l'onglet **Sortie**, MapForce évalue, pour chaque nœud expense-item, la condition connectée à l'entrée **bool** du filtre. Lorsque la condition est vraie, le nœud expense-item est passé sur le cible ; sinon, elle sera ignorée. Par conséquent, seuls les items de dépense correspondant aux critères seront affichés dans la sortie :

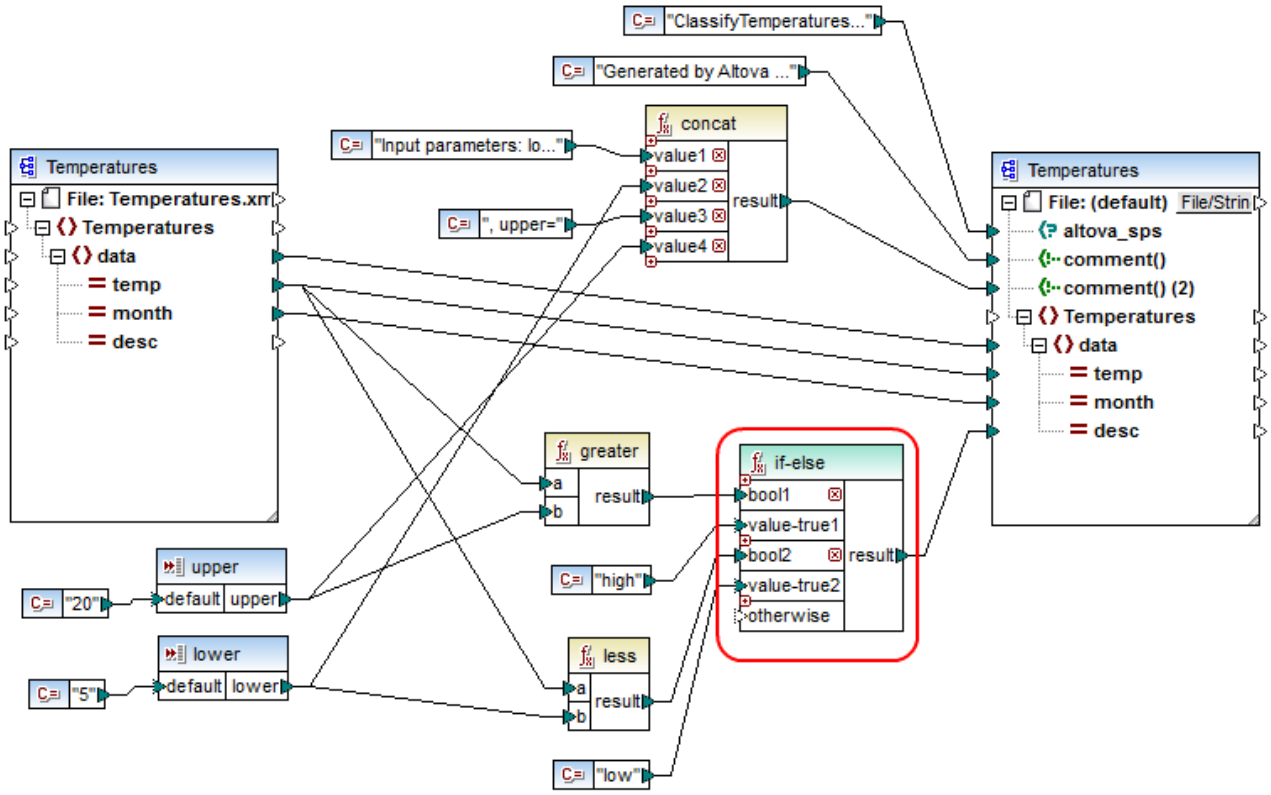
```
...
   <expense-item>
      <type>Meal</type>
      <Date>2003-01-01</Date>
      <expense>122.11</expense>
   </expense-item>
   <expense-item>
      <type>Lodging</type>
      <Date>2003-01-02</Date>
      <expense>299.45</expense>
   </expense-item>
...
```

Sortie XML après exécution du mappage

5.5.2 Exemple: Retourner une valeur par condition

Cet exemple vous montre comment retourner une valeur simple depuis un composant, basée sur une condition vraie/faux. Une **Condition If-Else** ( ) est utilisée pour atteindre l'objectif. Veuillez noter que les **Conditions If-Else** ne doivent pas être confondues avec les composants de filtre. Les **Conditions If-Else** sont uniquement applicables si vous souhaitez traiter des valeurs simples de manière conditionnelle (string, entier, etc.). Si vous souhaitez filtrer des valeurs complexes comme des nœuds, utiliser un filtre à la place (voir Exemple : Filtrer des nœuds 177).

Le mappage décrit dans cet exemple est disponible sous le chemin suivant : **<Documents>\Altova\MapForce2024\MapForceExamples\ClassifyTemperatures.mfd**.



Ce mappage lit des données depuis un XML de source qui contient des données de température ("Temperatures") et écrit des données vers un XML de cible XML qui se conforme au même schéma. Il existe plusieurs autres composants entre la cible et la source, un d'entre eux est la condition **if-else** (marquée en rouge), qui est aussi le sujet de cette rubrique.

L'objectif de ce mappage est d'ajouter une brève description à chaque enregistrement de température dans la cible. Spécifiquement, si la température se situe au-dessus de 20 degrés Celsius, la description doit être "high". Si la température se situe à 5 degrés Celsius, la description doit être "low". Pour tous les autres cas, aucune description ne doit être écrite.

Pour atteindre cet objectif, un traitement par condition est nécessaire ; c'est pourquoi une Condition If-Else a été ajoutée au mappage. (Pour ajouter une Condition If-Else, cliquer le menu **Insérer**, puis cliquer sur la **Condition If-Else**.) Dans ce mappage, la Condition If-Else a été étendue (avec l'aide de la touche ⊡) pour accepter deux conditions : **bool1** et **bool2**.

Les conditions elles-mêmes sont fournies par les fonctions **`greater`** et **`less`**, qui ont été ajoutées depuis la bibliothèque principale MapForce (pour plus d'informations, voir aussi Ajouter une fonction au mappage 195). Ces fonctions évaluent les valeurs fournies par deux composants d'entrée, appelés "upper" et "lower". (Pour ajouter un composant d'entrée, cliquer sur le menu **Insérer**, puis sur **Insérer entrée**. Pour plus d'informations à propos des composants d'entrée, voir Fournir des paramètres au mappage 147.

Les fonctions **`greater`** et **`less`** retournent soit true soit false. Le résultat de fonction détermine ce qui est écrit dans l'instance cible. Concrètement, si la valeur de l'attribut "temp" dans la source est supérieur à 20, la valeur de constante "high" est passée à la condition **if-else**. Si la valeur de l'attribut "temp" dans la source est

inférieure à 5, la valeur de constante "low" est passée dans la condition **if-else**. L'entrée **otherwise** n'est pas connectée. C'est pourquoi, si aucune des conditions ci-dessus n'est remplie, rien ne sera passé dans le connecteur de sortie **result**.

Enfin, le connecteur de sortie **result** fournit cette valeur (une fois pour chaque enregistrement de température) vers l'attribut "desc" dans la cible.

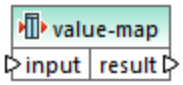
Une fois que vous êtes prêt à consulter le résultat de mappage, cliquer sur l'onglet **Sortie**. Veuillez noter que la sortie XML résultante contient maintenant l'attribut "desc", que la température soit supérieure à 20 ou inférieure à 5.

```
...
   <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"/>
...
```

Sortie XML après exécution du mappage

5.6 Value-Maps

Le composant Value-Map (*capture d'écran ci-dessous*) vous permet de transformer une valeur par une autre valeur à l'aide d'une table de consultation prédéfinie. Un tel composant ne traite qu'une seule valeur à la fois ; c'est pourquoi il a une **entrée** et un **résultat** dans le mappage.



Un Value-Map est utile lorsque vous souhaitez mapper des items individuels dans deux ensembles pour pouvoir remplacer des items. Par exemple, vous pourriez mapper les jours de la semaine exprimés en chiffres (1, 2, 3, 4, 5, 6 et 7) pour le nom de chaque jour de la semaine ("Lundi", "Mardi", etc). De même, vous pourriez mapper les noms des mois ("Janvier", "Février", "Mars", etc) dans la représentation numérique de chaque mois (1, 2, 3, etc). Au moment de l'exécution du mappage, les valeurs correspondantes seront remplacées conformément à votre table de consultation personnalisée. Les valeurs dans les deux ensembles peuvent être de types différents, mais chaque ensemble doit stocker des valeurs du même type de données.

Les composants Value-Map sont idéaux pour des look-ups simples, où chaque valeur dans le premier ensemble correspond à la valeur unique dans le deuxième ensemble. Si une valeur ne peut pas être trouvée dans la table de consultation, vous pouvez soit la remplacer avec une valeur personnalisée ou une valeur vide, soit la transmettre telle quelle. Si vous souhaitez consulter ou filtrer des valeurs sur la base de critères plus complexes, utiliser un des composants de filtre 176 à la place.

Plus important, lorsque vous générez du code ou que vous compilez un fichier d'exécution MapForce Server à partir du mappage, les données de la table de consultation sont intégrées dans le code ou le fichier généré. Par conséquent, la définition d'une table de consultation directement sur le mappage est un bon choix uniquement si vos données ne changent pas fréquemment et ne sont pas trop volumineuses (moins de quelques centaines d'enregistrements). Si les données changent fréquemment, il peut s'avérer difficile d'entretenir le mappage et le code généré régulièrement, il est plus simple d'entretenir les données de consultation en tant que texte, XML, base de données ou Excel.

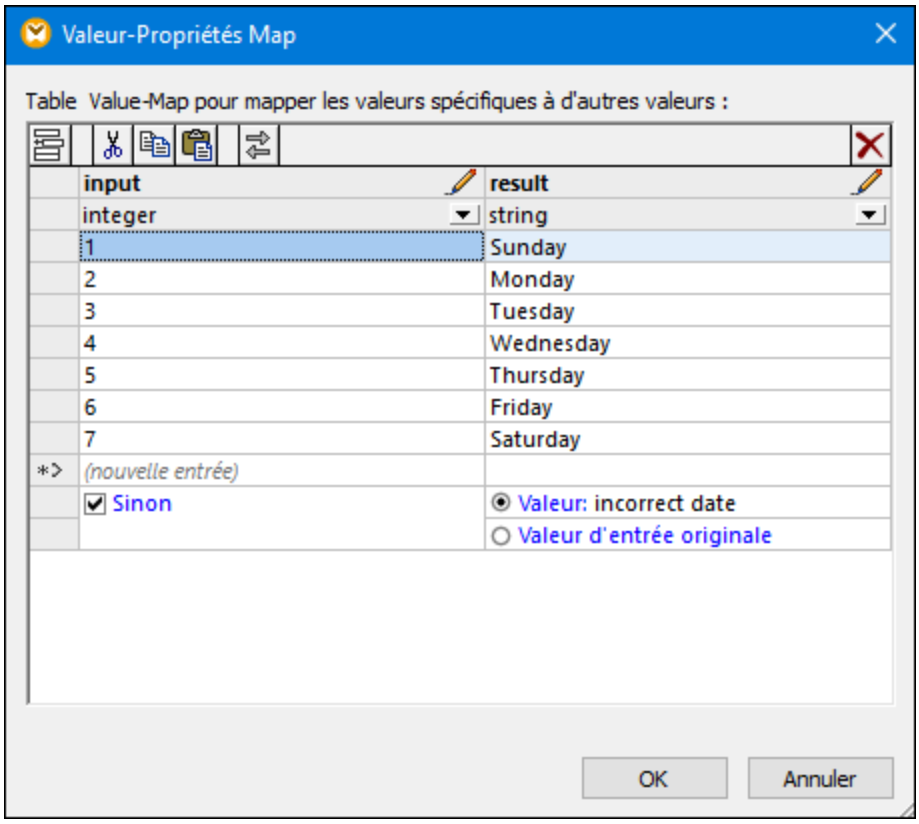
Si la table de consultation est très volumineuse, l'exécution de mappage sera ralentie par la table de consultation. Dans ce cas, il est recommandé d'utiliser un composant de base de données avec SQL-Where à la place. Les composants de base de données sont disponibles dans les éditions MapForce Professional et Enterprise. Les bases de données sont de bons candidats pour cela, en raison de leur portabilité. Du côté serveur, vous pouvez améliorer la performance des tables de consultation en exécutant un mappage avec MapForce Server ou MapForce Server Advanced Edition.

Créer une Value-Map

Pour ajouter un composant Value-Map au mappage, faites une des choses suivantes :

- Cliquer sur la touche de barre d'outils **Insérer Value-Map**  .
- Dans le menu **Insérer**, cliquer sur **Value-Map**.
- Cliquer avec la touche de droite sur une connexion, et sélectionner **Insérer Value-Map** depuis le menu contextuel.

Cela ajoute un nouveau composant Value-Map dans le mappage. Vous pouvez maintenant commencer à ajouter des paires d'items dans la table de consultation. À cette fin, double-cliquez sur le bouton droit sur la barre de titre du composant, et sélectionnez **Propriétés** depuis le menu contextuel.



Au moment de l'exécution du mappage, MapForce contrôle chaque valeur qui atteint l'**entrée** de Value-Map. S'il existe une valeur correspondante *dans la colonne gauche* de la table de consultation, alors la valeur d'entrée originale sera remplacée avec la valeur *provenant de la colonne de droite*. Sinon, vous pouvez la configurer en option pour retourner un des éléments suivants :

- Une valeur de remplacement. Dans l'exemple ci-dessus, la valeur de remplacement est le texte « incorrect date ». Vous pouvez également définir la valeur de remplacement pour qu'elle soit vide, en ne saisissant aucun texte.
- La valeur d'entrée originale. Cela signifie que, si aucune correspondance n'est trouvée dans la table de consultation, la valeur d'entrée originale sera transmise plus loin dans le mappage, sans être modifiée.

Si vous ne configurez pas une condition "Otherwise", le Value-Map retourne un **nœud vide** à chaque fois qu'une correspondance n'est pas trouvée. Dans ce cas, rien ne sera transmis dans le composant cible et la sortie contiendra des champs manquants. Pour éviter que cela ne se produise, vous devriez soit configurer la condition "Otherwise" soit utiliser la fonction **`substitute-missing`**[304].

Il existe une différence entre configurer une valeur de remplacement vide et ne pas spécifier la condition "Otherwise". Dans le premier cas, le champ sera généré dans la sortie, mais il aura une valeur vide. Dans le deuxième cas, le champ (ou l'élément XML) entourant la valeur ne sera pas créé du tout. Pour plus d'informations, voir Exemple : Remplacer les titres de tâches[190].

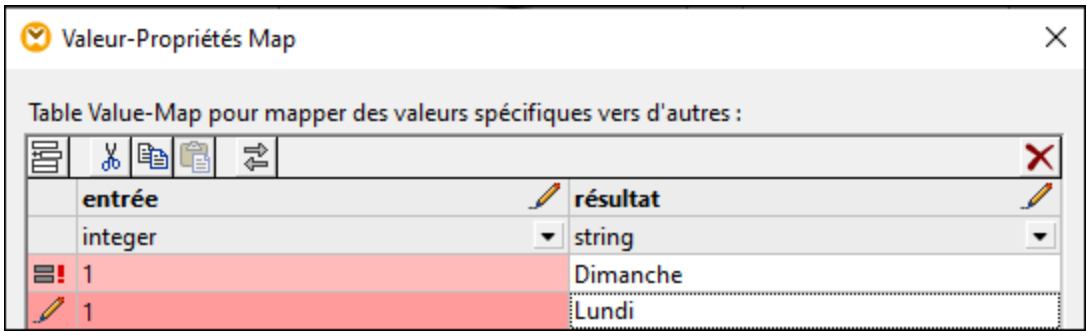
Remplir un Value-Map

Dans une table de consultation, vous pouvez définir autant de paires de valeurs que nécessaire. Vous pouvez soit saisir les valeurs manuellement, soit copier-coller des données tabulaires depuis des fichiers texte, CSV, ou Excel. Les tables Copier-coller depuis une page HTML utilisant un navigateur commun fonctionnera également dans la plupart des cas. Si vous copiez des données depuis des fichiers de texte, les champs doivent être séparés par des caractères de tabulateur. De plus, MapForce reconnaîtra du texte séparé par des virgules ou des points-virgules dans la plupart des cas.

Veuillez considérer les éléments suivants lorsque vous créez des tables de consultation :

1. Tous les items dans la colonne de gauche doivent être uniques. Autrement, il ne serait pas possible de déterminer quel item vous voulez faire correspondre spécifiquement.
2. Les items qui correspondent à la même colonne doivent être du même type de données. Vous pouvez choisir le type de données depuis la liste déroulante au sommet de chaque colonne dans la table de consultation. Si vous devez convertir les types booléens, saisissez le texte « true » ou « false » litéralement. Pour consulter une illustration de ce cas, voir Exemple : Remplacer les jours de la semaine 187.

Si MapForce rencontre des données invalides dans la table de consultation, il affiche un message d'erreur et souligne les lignes invalides en rose, par exemple :



Pour importer des données depuis une source externe dans le composant Value-Map, suivez les étapes suivantes :

1. Sélectionner les cellules pertinentes dans le programme de source (par exemple, Excel). Cela peut être soit une colonne de données simple ou deux colonnes adjacentes.
2. Copier des données dans le presse-papiers en utilisant la commande **Copier** du programme externe.
3. Dans le composant Value-Map, cliquer sur la ligne précédent celle dans laquelle vous souhaitez coller les données.
4. Cliquer sur la touche **Coller la table depuis le presse-papiers** dans le composant Value-Map. En alternative, appuyez sur **Ctrl+V** ou **Shift+Insert**.

Note : La touche **Coller la table depuis le presse-papiers** est uniquement activée si vous avez tout d'abord copié des données depuis une source (donc, si des données se trouvent dans le presse-papiers).

Lorsque vous données contenues dans le presse-papiers contient plusieurs colonnes, alors seules les données provenant des premières deux colonnes sont insérées dans la table de consultation ; toutes les autres colonnes seront ignorées. Si vous collez des données provenant d'une seule colonne sur des valeurs existantes, un menu contextuel apparaît, vous demandant si les données contenues dans le presse-papiers

doivent être insérées dans les nouvelles lignes ou si les lignes existantes doivent être écrasées. Ainsi, si vous souhaitez écraser les valeurs existantes dans la table de consultation (par opposition à l'insertion de nouvelles lignes), veuillez vous assurer que le presse-papiers ne contient qu'une seule colonne.

Pour insérer des lignes manuellement avant une ligne existante, cliquer tout d'abord sur la ligne concernée, puis cliquer sur la touche **Insertion** .

Pour déplacer une ligne existante à une autre position, glisser la ligne sur la nouvelle position (vers le haut ou vers le bas) tout en maintenant appuyée la touche de gauche de la souris.

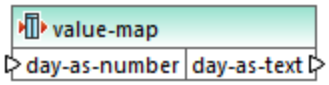
Pour copier ou couper des lignes pour les coller ensuite sur une autre position, sélectionner tout d'abord la ligne, puis cliquer sur la touche **Copier** (ou sur la touche **Couper** , respectivement). Vous pouvez aussi copier ou couper plusieurs lignes qui ne sont pas forcément consécutives. Pour sélectionner plusieurs lignes, maintenir appuyée la touche **Ctrl** tout en cliquant sur la ligne. Veuillez noter que le texte coupé ou copié contient toujours des valeurs provenant des deux colonnes ; vous ne pouvez pas couper ou copier des valeurs provenant d'une seule colonne.

Pour supprimer une ligne, cliquer dessus puis cliquer sur la touche **Supprimer** .

Pour changer l'ordre de deux colonnes, cliquer sur la touche **Échanger** .

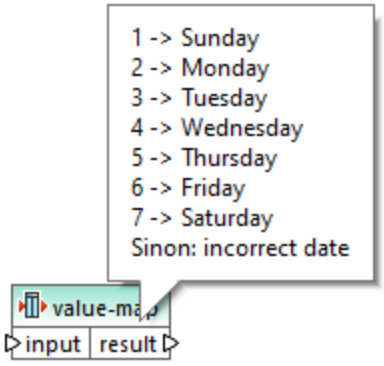
Renommer des paramètres de Value-Map

Par défaut, le paramètre d'entrée d'un composant Value-Map est appelé "entry" et le paramètre de sortie est appelé "result". Pour rendre le mappage plus clair, vous pouvez renommer en option un de ces paramètres en cliquant sur la touche **Édition** située à côté du nom respectif. Le graphique suivant est un exemple d'un Value-Map avec des noms de paramètre personnalisés :



Préconsulter un Value-Map

Une fois le Value-Map créé, vous pouvez rapidement consulter sa mise en place directement dans le mappage en plaçant la souris au-dessus de la barre de titre du composant :

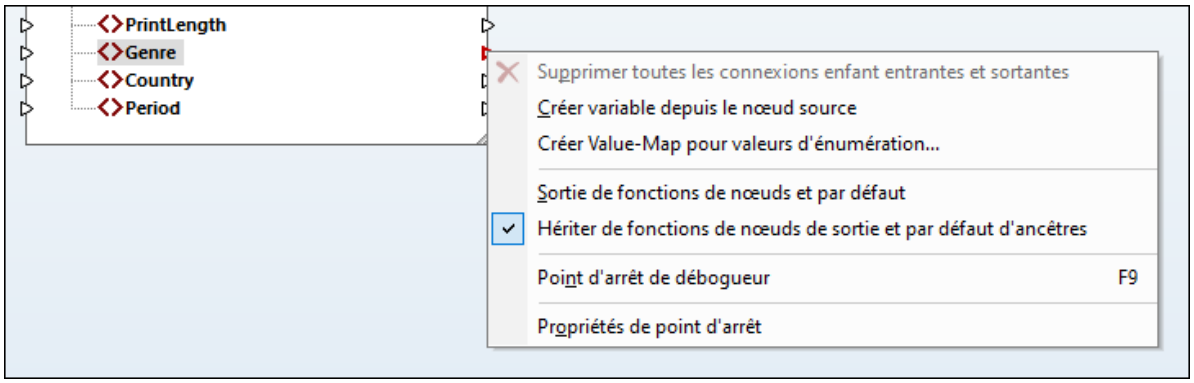


Créer un Value-Map depuis le type d'énumération

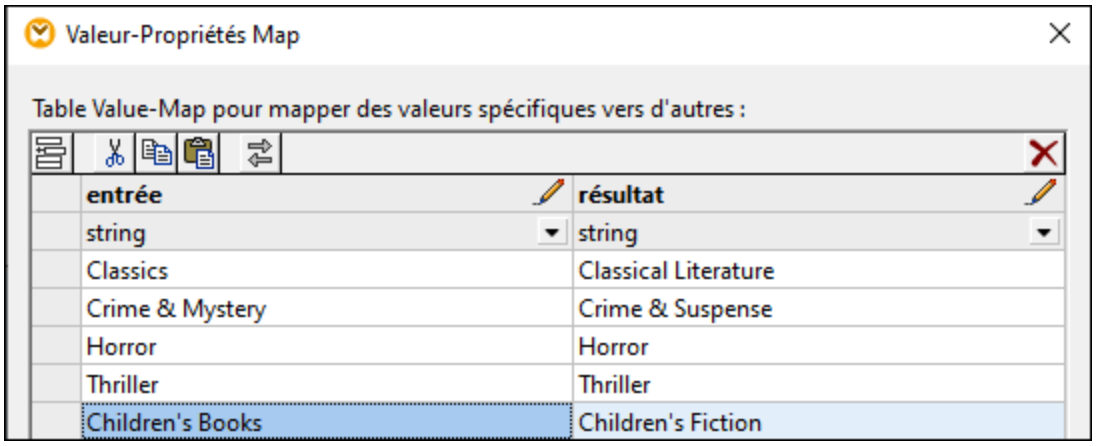
MapForce vous permet de créer un Value-Map depuis de nœuds avec des valeurs d'énumération. Cette fonction est actuellement prise en charge pour les composants XML dont les nœuds ont des facettes d'énumération (*toutes éditions*) et composants EDI dont les nœuds ont des listes de code EDI (*Enterprise Edition*). Dépendant de vos besoins, vous pouvez créer un tel Value-Map depuis le connecteur d'entrée ou de sortie du nœud.

Pour créer un Value-Map à partir de type d'énumération, suivez les instructions ci-dessous :

1. Dépendant de vos objectifs, cliquez avec la touche de droite ou connecteur de sortie du nœud pour les valeurs d'énumération pour lesquelles vous souhaitez créer un Value-Map. Dans notre exemple (*capture d'écran ci-dessous*), nous avons sélectionné le connecteur de sortie du nœud `Genre`.



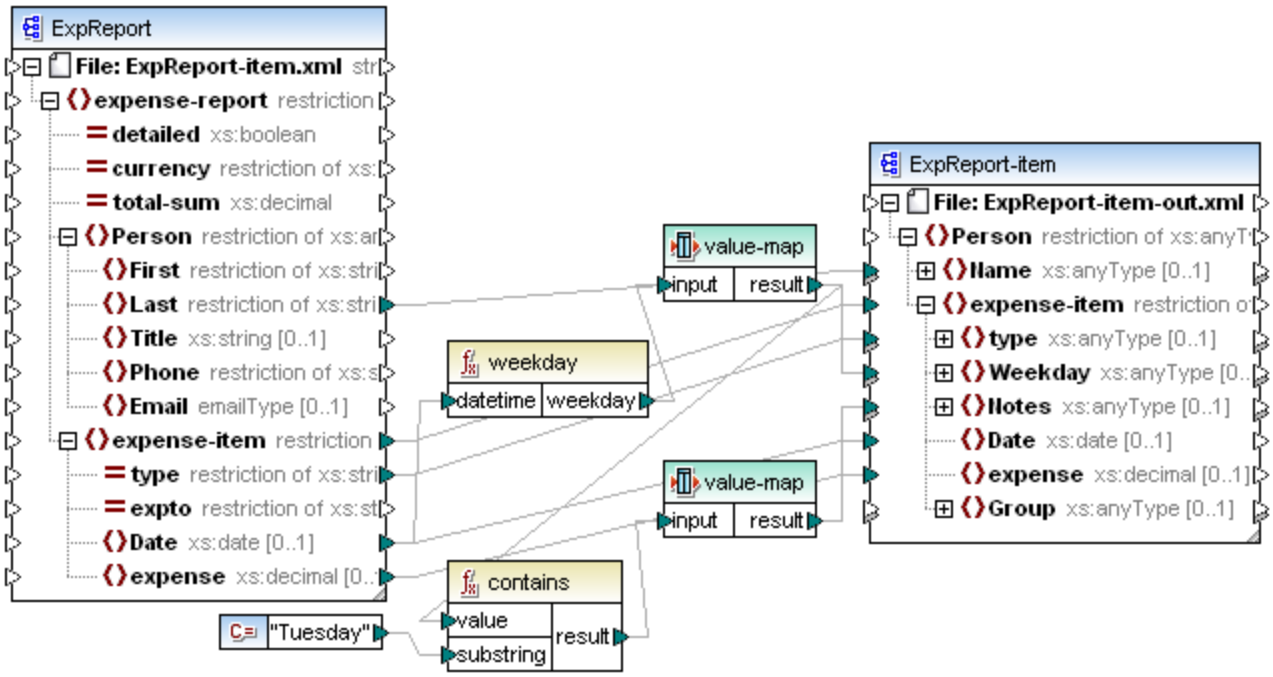
2. Sélectionnez **Créer Value-Map pour Valeurs d'énumération** depuis le menu contextuel.
3. Le dialogue **Propriétés Value-Map** apparaît. Les deux parties `input` et `result` de Value-Map sont pré-remplies avec les mêmes valeurs d'énumération. Pour pouvez revoir et éditer les valeurs tel que requis. La capture d'écran ci-dessous affiche la liste des valeurs `Genre` inhérente au fichier XML source d'origine (`input`) et la liste des valeurs modifiées que nous voulons mapper (`result`).



4. Après avoir revu les valeurs d'énumération, cliquez sur **OK**. Cette action ajoutera un composant Value-Map au domaine de mappage. Le composant Value-Map sera automatiquement connecté au nœud dont les valeurs d'énumération du Value-Map ont été créées.
5. Connectez les autres paramètres du Value-Map avec le nœud pertinent et continuez de concevoir votre mappage tel que requis.

5.6.1 Exemple : Remplacer les jours de la semaine

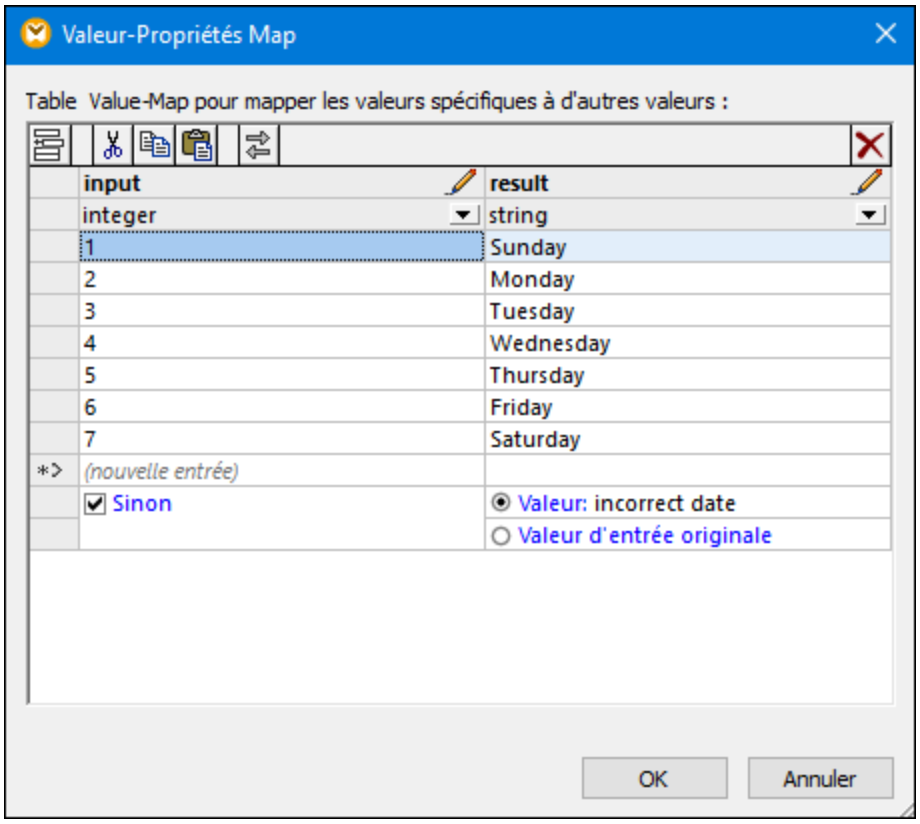
Cet exemple illustre un Value-Map qui remplace des valeurs entières avec les noms du jour de la semaine (1 = Dimanche, 2 = Lundi, etc.). Cet exemple est accompagné par un mappage qui est disponible dans le chemin suivant : **<Documents>\Altova\MapForce2024\MapForceExamples\Tutorial\Expense-valmap.mfd**.



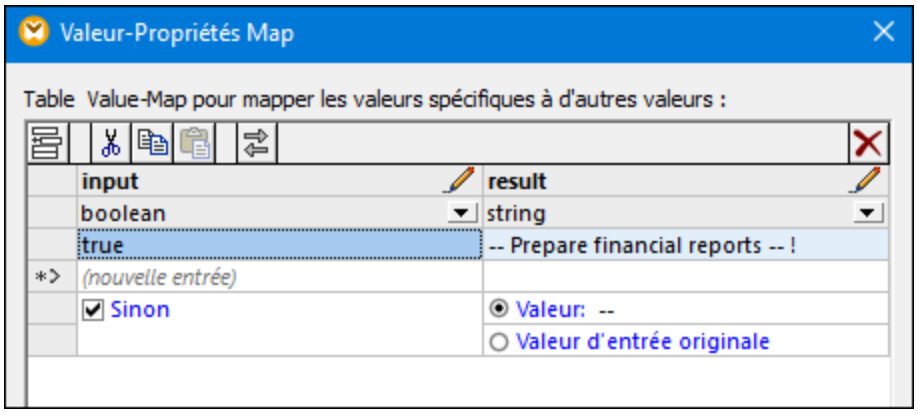
Expense-valmap.mfd

Ce mappage extrait le jour de la semaine depuis l'item **Date** dans le fichier de source, convertit la valeur numérique dans le texte et l'écrit dans l'item **Weekday** du composant de cible. Plus spécifiquement, la chose suivante se produit :

- La fonction **weekday** extrait le numéro du jour de la semaine depuis l'item **Date** dans le fichier de source. Le résultat de cette fonction sont des entiers allant de 1 à 7.
- Le premier composant Value-Map transforme les entiers dans les jours de la semaine (1 = Dimanche, 2 = Lundi, etc.). Si le composant rencontre un entier invalide en-dehors de la plage 1-7, il retournera le texte "incorrect date".



- Si le jour de la semaine contient "Tuesday", alors le texte "Prepare Financial Reports" sera écrit dans l'item **Notes** dans le composant de cible. Cela s'effectue avec l'aide de la fonction **`contains`**, qui passe une valeur booléenne **true** ou **false** dans un second composant Value-Map. Le second Value-Map présente la configuration suivante :



Le Value-Map illustré ci-dessus doit être compris comme suit :

- Lorsqu'une booléenne **true** est rencontrée, la convertir dans le texte "-- Prepare financial reports -- ! ". Pour tous les autres cas, elle retourne le texte "--".

Veuillez noter que le type de données de la première colonne est définie sur "boolean". Cela garantit que la valeur booléenne d'entrée **true** est reconnue comme telle.

5.6.2 Exemple : Remplacer des titres de tâche

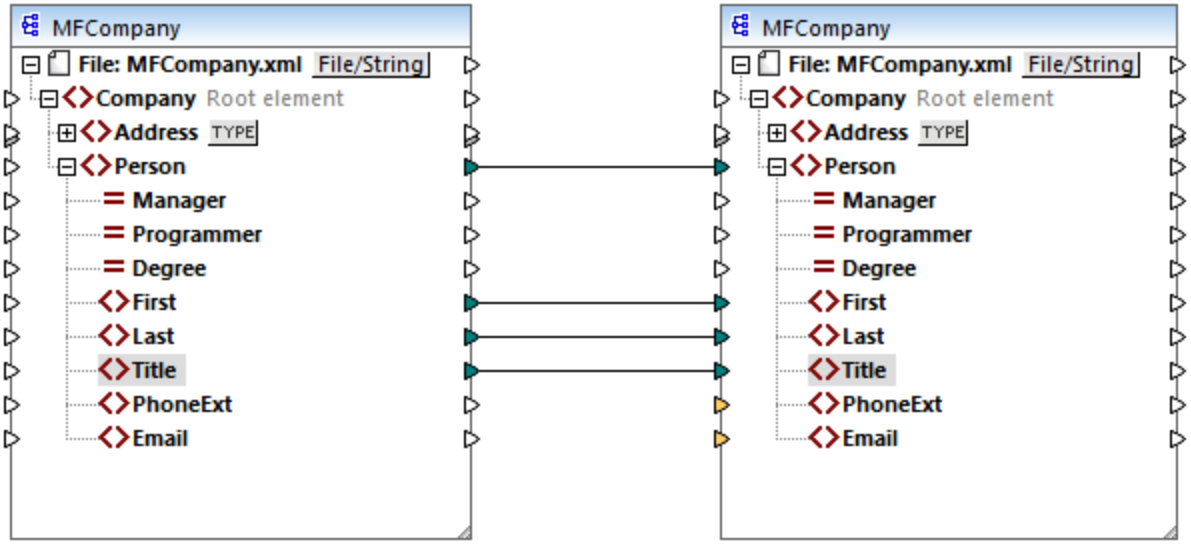
Cet exemple vous montre comment remplacer des valeurs d'éléments spécifiques dans un fichier XML à l'aide de composants Value-Map (c'est à dire, en utilisant une table de consultation prédéfinie).

Le fichier XML nécessaire pour cet exemple est disponible dans le chemin suivant : **<Documents>\Altova\MapForce2024\MapForceExamples\Tutorial\ MFCompany.xml**. Il stocke, entre autres données, des informations concernant les employés de l'entreprise et leurs titres de poste, par exemple :

```
<Person>
   <First>Michelle</First>
   <Last>Butler</Last>
   <Title>Software Engineer</Title>
</Person>
<Person>
   <First>Lui</First>
   <Last>King</Last>
   <Title>Support Engineer</Title>
</Person>
<Person>
   <First>Steve</First>
   <Last>Meier</Last>
   <Title>Office Manager</Title>
</Person>
```

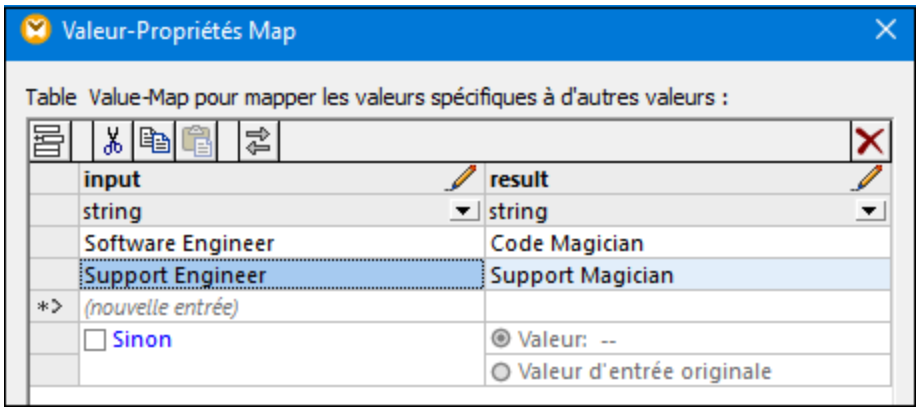
Partons du principe que vous souhaitez remplacer certains des titres de poste dans le fichier XML ci-dessus. En particulier, le titre "Software Engineer" doit être remplacé par "Code Magician". De même, le titre "Support Engineer" doit être remplacé par "Support Magician". Les autres titres de poste ne doivent pas être modifiés.

Pour atteindre cet objectif, ajouter le fichier XML à la surface de mappage, en cliquant sur la touche de la barre d'outils **Insérer Schéma/Fichier XML**  ou en exécutant la commande de menu **Insérer | Schéma/Fichier XML**. Ensuite, copier-coller le composant XML dans le mappage et créer les connexions comme indiqué ci-dessous. Veuillez noter que vous devrez éventuellement tout d'abord désactiver l'option dans la barre d'outils  **Basculer l'auto-connexion des enfants**, pour pouvoir éviter l'établissement automatique de connexions inutiles.



Le mappage créé jusqu'à présent copie les éléments **Person** dans le fichier cible XML, sans effectuer de modifications dans les éléments **First**, **Last** et **Title**.

Pour remplacer les titres de postes requis, ajoutons un composant Value-Map. Cliquer avec la touche de droite sur la connexion entre les deux éléments **Title**, et choisir **Insérer Value-Map** depuis le menu contextuel. Configurer les propriétés Value-Map comme indiqué ci-dessous :

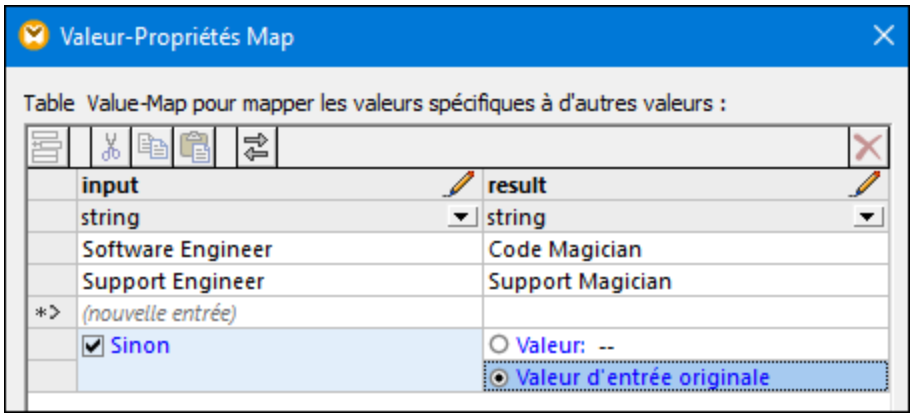


Conformément à la configuration ci-dessus, chaque occurrence de "Software Engineer" sera remplacée par "Code Magician", et chaque occurrence de "Support Engineer" sera remplacée par "Support Magician". Veuillez noter que la condition **Otherwise** n'est pas encore spécifiée. C'est pour cette raison que Value-Map retourne un *nœud vide* à chaque fois que le titre de poste est différent de "Software Engineer" et de "Support Engineer". Par conséquent, si vous cliquez sur l'onglet **Sortie** et que vous consultez le mappage, certains des éléments **Person** auront un titre manquant **Title**, par exemple :

```
<Person>
   <First>Vernon</First>
   <Last>Callaby</Last>
</Person>
```

```
<Person>
   <First>Frank</First>
   <Last>Further</Last>
</Person>
<Person>
   <First>Michelle</First>
   <Last>Butler</Last>
   <Title>Code Magician</Title>
</Person>
```

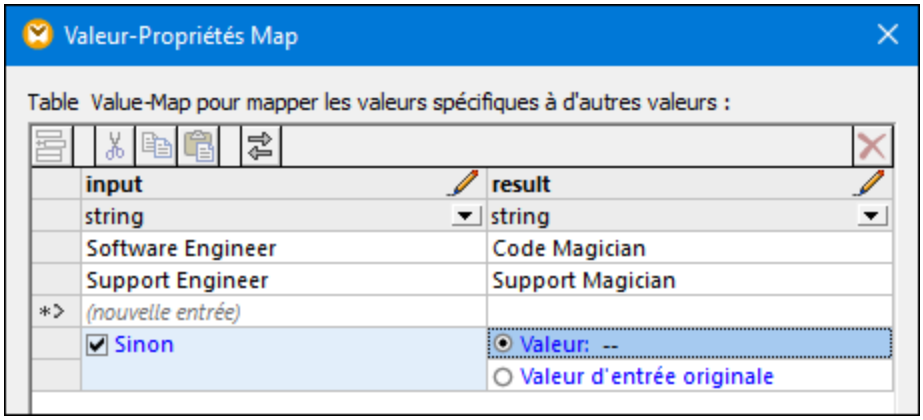
Comme indiqué auparavant, des nœuds vides entraînent des entrées manquantes dans la sortie générée ; c'est pourquoi, dans le fragment XML ci-dessus, seul le titre de Michelle Butler a été remplacé, étant donné que son titre était présent dans la table de consultation. La configuration créée jusqu'à présent ne remplit toujours pas l'exigence originale. La configuration correcte est la suivante :



Avec la configuration ci-dessus, la chose suivante se produit au moment de l'exécution :

- Chaque occurrence de "Software Engineer" sera remplacée par "Code Magician"
- Chaque occurrence de "Support Engineer" sera remplacée par "Support Magician"
- Si le titre original ne se trouve pas dans la table de consultation, le Value-Map le retournera sans le modifier.

À seul titre illustratif, nous pouvons aussi changer tous les titres de poste différents de "Software Engineer" et de "Support Engineer" en une valeur personnalisée, par exemple "N/A". Pour ce faire, définir les propriétés Value-Map comme indiqué ci-dessous :



Maintenant, lorsque vous consultez le mappage, chaque titre de poste est présent dans la sortie, mais ceux qui n'avaient pas de correspondance auront la valeur "N/A", par exemple :

```
<Person>
   <First>Vernon</First>
   <Last>Callaby</Last>
   <Title>N/A</Title>
</Person>
<Person>
   <First>Frank</First>
   <Last>Further</Last>
   <Title>N/A</Title>
</Person>
<Person>
   <First>Michelle</First>
   <Last>Butler</Last>
   <Title>Code Magician</Title>
</Person>
```

Cela complète l'exemple Value-Map. En appliquant la logique ci-dessus, vous pouvez obtenir le résultat désiré dans d'autres mappages.

6 Fonctions

Dans MapForce, vous pouvez utiliser les catégories de fonction suivantes pour transformer des données conformément à vos besoins :

- **Fonctions intégrées MapForce** — ces fonctions sont prédéfinies dans MapForce et vous pouvez les utiliser dans vos mappage pour effectuer une gamme très étendue de tâches de traitement impliquant des strings, des nombres, des dates, et d'autres types de données. Vous pouvez aussi les utiliser pour effectuer des regroupements, des agrégations, des numérotations automatiques ainsi que d'autres tâches. Pour consulter une référence à toutes les fonctions intégrées disponibles, voir Référence des bibliothèques de fonctions 232.
- **Fonctions définies par l'utilisateur (FDU)** — il s'agit de fonctions MapForce que vous pouvez créer vous-même, en utilisant en tant que base les types de composants natifs et les fonctions intégrées déjà disponibles dans MapForce, voir Fonctions définies par l'utilisateur 203.
- **Fonctions personnalisées** — il s'agit des fonctions que vous pouvez importer depuis des sources externes comme des bibliothèques XSLT et les adapter à MapForce. Veuillez noter, afin de pouvoir les réutiliser dans MapForce, que vos fonctions personnalisées doivent retourner des données de type simple (comme des string ou des entiers) et elles doivent aussi prendre des paramètres de type simple. Pour plus d'informations, voir Importer des fonctions XSLT personnalisées 220

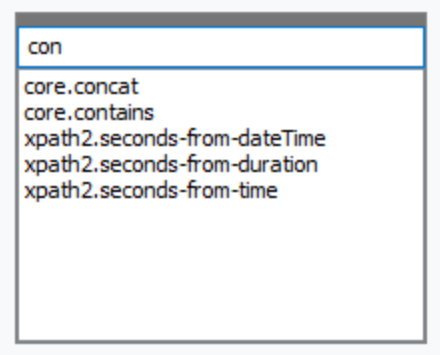
6.1 Notions fondamentales de base

Les sous-sections ci-dessous donnent un aperçu des actions liées à la fonction de base. Les fonctions que vous voyez dans la fenêtre des Bibliothèques dépendent du langage de transformation que vous avez sélectionné. Pour plus d'information, voir Langages de transformation 17.

Ajouter une fonction

MapForce inclut un grand nombre de fonctions built-in que vous pouvez ajouter au mappage. Pour plus d'informations concernant toutes les fonctions intégrées disponibles, voir Référence des bibliothèques de fonctions 232. Pour ajouter une fonction au mappage, vous pouvez choisir une des méthodes suivantes :

- Appuyez et gardez appuyer la fonction requise dans la fenêtre Bibliothèques et glissez-la dans la zone de mappage. Pour filtrer les fonctions par leur nom, commencez à saisir le nom de la fonction dans le champ de texte situé en bas de la fenêtre.
- Double-cliquez où vous voulez dans la zone vide du mappage et commencez à saisir le nom de la fonction (*voir la capture d'écran ci-dessous*). Pour voir une info-bulle avec plus de détails concernant une fonction, choisir une fonction parmi la liste. Pour ajouter une fonction à votre mappage, double-cliquez sur la fonction pertinente dans la zone de liste déroulante.



Ajouter une UDF

Vous pouvez aussi ajouter des fonctions définies par l'utilisateur (UDF) à votre mappage en utilisant les mêmes approches que décrites ci-dessous si (i) une UDF a déjà été créée dans le même mappage ou (ii) si vous avez importé le mappage qui contient une UDF comme bibliothèque locale ou globale.

Ajouter une constante

Les constantes vous permettent de fournir du texte personnalisé et les nombres au mappage. Pour ajouter une constante au mappage, vous pouvez choisir une des options suivantes :

- Cliquez avec la touche de droite sur la zone de mappage vide et sélectionnez **Insérer constante** depuis le menu contextuel. Saisissez la valeur et sélectionnez un des types de données : *string*, *nombre* ou *tous les autres*.
- Sélectionnez la commande de menu **Insérer | Constante**. Saisissez la valeur et sélectionnez un des types de données : *string*, *nombre* ou *tous les autres*.
- Cliquez sur la commande **Constante** de la barre d'outils. Saisissez la valeur et sélectionnez un des types de données : *string*, *nombre* ou *tous les autres*.
- Double-cliquer où vous voulez dans une zone de mappage vide. Saisissez les doubles guillemets anglais suivis de la valeur constante. Le double guillemet anglais de fermeture est optionnel. Pour ajouter une constante numérique, il suffit de saisir le nombre.

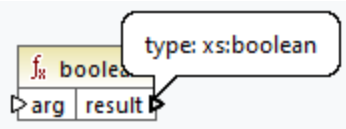
Chercher une fonction

Pour rechercher une fonction dans la fenêtre **Bibliothèques**, commencez à saisir le nom de la fonction dans le champ de texte en bas de la fenêtre. Par défaut, MapForce effectue la recherche par le biais du nom de fonction et du texte de description. Si vous voulez exclure la description de fonction de la recherche, cliquez sur la flèche déroulante vers le bas et désactivez l'option *Recherche dans les descriptions de fonction*. Pour annuler la recherche, appuyer sur la touche **Esc** ou cliquez sur ✖.

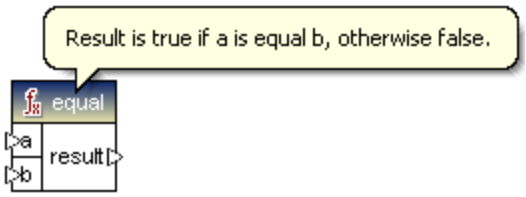
Pour trouver toutes les occurrences d'une fonction dans le mappage actif actuellement, cliquez avec la touche de droite sur le nom de la fonction dans la fenêtre **Bibliothèques** et choisissez **Trouver tous les appels** depuis le menu contextuel. Les résultats de recherche sont affichés dans la fenêtre **Messages**.

Consulter un type et une description de fonction

Pour consulter le type de données d'un argument d'entrée/de sortie, déplacez votre souris au-dessus de la partie argument d'une fonction (*voir la capture d'écran ci-dessous*). Veuillez vous assurer que la touche de la barre d'outils (**Afficher astuces** est activée.

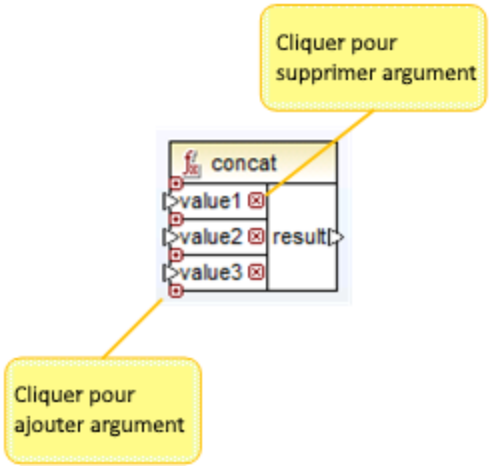


Pour afficher la description d'une fonction, déplacez le curseur au-dessus de l'en-tête de la fonction (*voir la capture d'écran ci-dessous*). Veuillez vous assurer que la touche de la barre d'outils (**Afficher astuces** est activée.



Ajouter/supprimer les arguments de fonction

Pour quelques fonctions intégrées de MapForce, il est possible d'ajouter autant de paramètres dont vous avez besoin pour vos objectifs de mappage. Un de ces exemples est la fonction **concat** 307. Pour ajouter ou supprimer les arguments de fonction (pour les fonctions qui les prennent en charge), cliquez sur **Ajouter paramètre** (⊡) ou **Supprimer paramètre** (⊠) à côté du paramètre que vous voulez ajouter ou supprimer respectivement (*voir ci-dessous*). Le fait de déposer une connexion dans le symbole ⊡ ajoute le paramètre et le connecte.


Cliquer pour
supprimer argument
concat
value1
value2
value3
result
Cliquer pour
ajouter argument

6.2 Gérer les bibliothèques de fonction

Dans MapForce, vous pouvez importer et utiliser les types de bibliothèques suivants dans un mappage :

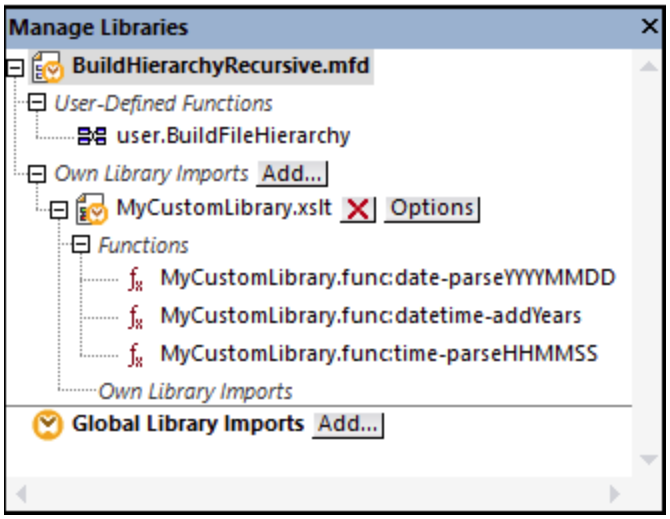
- Tout fichier design de mappage (*.mfd) qui contient les fonctions définies par l'utilisateur (UDF). Cela réfère spécifiquement aux fichiers de mappage qui contient des Fonctions définies par l'utilisateur (UDF) créés avec MapForce, en utilisant les fonctions intégrées MapForce et les composants en tant que blocs de construction. Pour plus d'information, voir Créer des fonctions définies par l'utilisateur 204.
- Les fichiers XSLT personnalisés qui contiennent des fonctions. Cela se réfère à des fonctions XSLT écrites en dehors de MapForce qui se qualifient pour l'importation dans MapForce comme décrit dans Importer des fonctions personnalisées XSLT 1.0 ou 2.0 220.

Gérer la fenêtre Bibliothèques

Vous pouvez consulter et gérer toutes les bibliothèques utilisées par un fichier de mappage provenant de la fenêtre Gérer des bibliothèques. Cela comprend les UDF et des bibliothèques personnalisées.

Par défaut, la fenêtre Gérer des bibliothèques n'est pas visible. Pour l'afficher, choisir une des deux options suivantes :

- Dans le menu **View**, cliquer sur **Gérer Bibliothèques**.
- Cliquer **Ajouter/Supprimer Bibliothèques** en bas de la fenêtre Bibliothèques.



Vous pouvez choisir de consulter des UDF et des bibliothèques uniquement pour le document de mappage (fichier .mfd) qui est activé actuellement, ou pour tous les mappages ouverts. Pour consulter des fonctions et des bibliothèques importées pour tous les documents de mappages ouverts actuellement, cliquer avec la touche de droite dans la fenêtre et sélectionner **Afficher les documents ouverts** depuis le menu contextuel.

Pour afficher le chemin du document de mappage ouvert au lieu du nom, cliquer avec la touche de droite dans la fenêtre et sélectionner **Afficher les chemins de fichier** depuis le menu contextuel.

Les données affichées dans la fenêtre Gérer Bibliothèques sont organisées en tant que hiérarchie arborescente comme suit :

- Tous les documents de mappage ouverts actuellement sont affichés en tant qu'entrées de niveau supérieur. Chaque entrée a deux branches: **Fonctions définies par l'utilisateur** et **Importations Bibliothèques propres**.
  - La branche **Fonctions définies par l'utilisateur** affiche des UDF contenues dans ce document.
  - La branche **Importations Bibliothèques propres** affiche des bibliothèques importées *localement* dans le document du mappage actuel. Le terme "bibliothèques" signifie d'autres documents de mappage (fichiers .mfd qui contiennent des fonctions définies par l'utilisateur) ou des bibliothèques externes personnalisées écrites en XSLT 1,0 XSLT 2.0, XQuery 1.0*, Java*, C#*, ou des fichiers .mff mentionnés précédemment. Notez que la structure **Importations de bibliothèque propre** pourrait se trouver à plusieurs niveaux profonds, puisque tout document de mappage peut importer tout autre document de mappage en tant que bibliothèque.
- La saisie **Global Library Imports** encadre les bibliothèques personnalisées que vous avez importées *globalement* au niveau de l'application. Là aussi, dans le cas de fichiers .mfd, la structure pourrait contenir plusieurs niveaux profonds pour les raisons mentionnées ci-dessus.

** Ces langages sont pris en charge uniquement dans les éditions Professional ou Enterprise de MapForce.*

Note : Les bibliothèques XSLT, XQuery, C# et Java peuvent avoir leur propres dépendances. Ces dépendances ne sont pas affichées dans la fenêtre Bibliothèques.

Commandes de menu contextuel

Vous pouvez effectuer rapidement plusieurs opérations par rapport à des objets dans la fenêtre Gérer des bibliothèques en cliquant avec la touche de droite sur un objet et en sélectionnant une des options de menu contextuel suivantes :

Commande	Description	Applicable pour
Ouvrir	Ouvre le mappage.	Mappages
Ajouter	Ouvre un dialogue dans lequel vous pouvez chercher une bibliothèque personnalisée des fonctions.	Importations Bibliothèque propre
Localiser une fonction dans la fenêtre Bibliothèques	Change le focus sur la fenêtre Bibliothèques et choisir la fonction.	Fonctions
Couper, Copier, Supprimer	Ces commandes standard de Windows sont uniquement applicables aux fonctions définies par l'utilisateur de MapForce. Vous ne pouvez pas copier-coller les fonctions de fichiers XSLT externes ou d'autres genres de bibliothèques.	Fonctions définies par l'utilisateur
Coller	Vous permet de coller une fonction définie par l'utilisateur qui a été copiée précédemment dans le presse-papiers à l'intérieur de la bibliothèque actuelle.	Bibliothèques (UDF)
Options	Ouvre une boîte de dialogue où vous pouvez définir ou changer des options pour la bibliothèque actuelle.	Bibliothèques

Commande	Description	Applicable pour
Afficher tous les Documents ouverts	Lorsque cette option est activée, la fenêtre Gérer les Bibliothèques affichera tous les mappages ouverts actuellement. Cela est généralement utile si vous souhaitez copier-coller des fonctions entre les mappages. Sinon, seul le mappage actuellement en cours est affiché.	Toujours
Afficher les chemins de fichier	Lorsque cette option est activée, les objets contenus dans la fenêtre Gérer les Bibliothèques sont affichés avec leur chemin complet. Sinon, seul le nom de l'objet est affiché.	Toujours

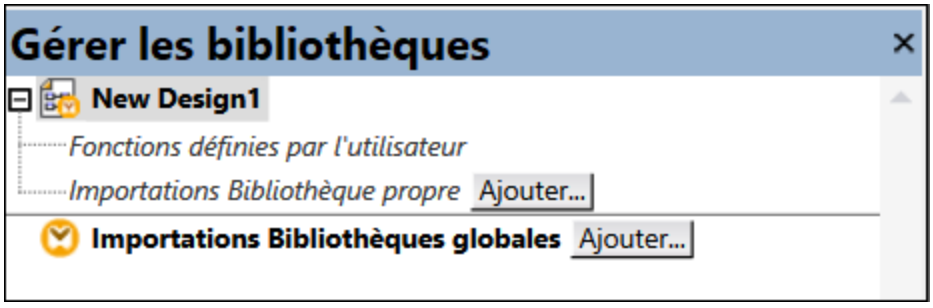
6.2.1 Bibliothèques locales et globales

Vous pouvez importer des bibliothèques *localement* ou *globalement*. Les importations globales ont lieu au niveau de l'application. Si une bibliothèque a été importée globalement, vous pouvez utiliser ses fonctions depuis n'importe quel mappage.

Les importations locales ont lieu au niveau du fichier du mappage. Par exemple, supposons que vous êtes en train de travailler sur le mappage **A.mfd**, et que vous décidez d'importer toutes les fonctions définies par l'utilisateur du mappage **B.mfd**. Dans ce cas, le mappage **B.mfd** est considéré être importé en tant que bibliothèque locale dans **A.mfd** et vous pouvez utiliser les fonctions provenant de **B.mfd** dans **A.mfd** également. De même, si vous importez des fonctions depuis un fichier XSLT dans **A.mfd**, il s'agit également d'une importation locale.

Vous pouvez consulter et gérer toutes les importations locales et globales depuis la fenêtre Gérer des bibliothèques. Pour importer des bibliothèques, suivre une des étapes suivantes :

1. Cliquez sur la touche **Ajouter/Supprimer des bibliothèques** en bas de la fenêtre Bibliothèques 21. La fenêtre **Gérer les Bibliothèques** s'ouvre (*voir la capture d'écran ci-dessous*).



2. Pour importer des fonctions en tant que bibliothèque *locale* (uniquement dans le cadre du document actuel), cliquez sur **Ajouter** sous le nom actuel du mappage. Pour importer les fonctions en tant que bibliothèque *globale* (au niveau du programme), cliquez sur **Ajouter** à côté des **Importations Bibliothèques globales**. Lorsque vous importez une bibliothèque *localement*, vous pouvez définir le chemin du fichier de bibliothèque pour qu'il soit relatif au fichier de mappage. Avec des bibliothèques importées globalement, le chemin de la bibliothèque importée est toujours absolu.

Noms de fonction conflictuels

Vous pouvez être confronté à des situation dans lesquelles le même nom de fonction est défini dans un des niveaux suivants :

- dans le mappage principal
- dans une bibliothèque qui a été importée localement
- dans une bibliothèque qui a été importée globalement

Lorsque ce genre de cas survient, MapForce tentera d'appeler la fonction exactement dans l'ordre ci-dessus, pour éviter toute ambiguïté. C'est à dire, la fonction définie directement dans le mappage prend précédence si le même nom de fonction existe dans une bibliothèque importée localement. De même, la fonction importée localement prend précédence sur la fonction importée globalement (en admettant que les deux fonctions portent le même nom).

S'il existe plusieurs fonctions portant le même nom, seule la fonction "vainqueur" sera appelée, conformément à la règle ci-dessus ; tout autre nom de fonction ambigu sera bloqué. Ces fonctions bloquées apparaissent grisées dans la fenêtre Bibliothèques, et il n'est pas possible de les utiliser dans le mappage.

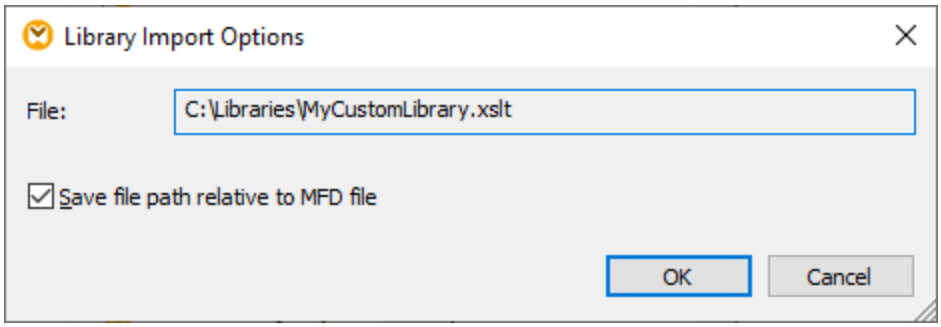
6.2.2 Chemins de bibliothèque relatifs

Vous pouvez définir le chemin de tout fichier de bibliothèque importé pour qu'il soit relatif au fichier de design du mappage (fichier .mfd), si la bibliothèque a été importée localement (pas globalement), tel que décrit dans Bibliothèque Locales et Globales 200.

> La configuration d'un chemin de bibliothèque relatif est uniquement applicable pour les bibliothèques qui ont été importées *localement* au niveau du document. Si un mappage a été importé *globalement* au niveau du programme, son chemin est toujours absolu.

Pour définir un chemin de bibliothèque comme relatif par rapport au fichier de design du mappage :

1. Cliquer sur **Ajouter/Supprimer des bibliothèques** en bas de la fenêtre Bibliothèques. La fenêtre de gestion des bibliothèques 23 s'ouvre.
2. Cliquer sur **Options** à côté de la bibliothèque qui vous intéresse. (En alternative, cliquer avec la touche de droite sur la bibliothèque, et sélectionner **Options** depuis le menu contextuel.)



3. Cocher la case à cocher **Enregistrer chemin de fichier en tant que relatif au fichier MFD**.

Note : si la case à cocher est grisée, veuillez vous assurer que la bibliothèque a bien été importée localement et pas globalement.

Si la case à cocher est cochée, MapForce tentera de suivre et de mettre à jour le chemin à tout fichier de bibliothèque référencé lorsque vous enregistrez le mappage fichier dans un nouveau répertoire en utilisant la commande de menu **Enregistrer sous**. De même, si les fichiers de bibliothèque se trouvent dans le même répertoire que le fichier de mappage, le chemin de référence ne sera pas cassé si vous déplacez tout le répertoire vers un autre emplacement sur le disque, voir aussi Utiliser des chemins relatifs sur un composant 41.

Veuillez noter que la case à cocher **Enregistrer chemin de fichier comme relatif sous le fichier MFD** spécifie que des chemins sont *relatifs au fichier de mappage*, et qu'elle n'affecte pas les chemins dans le code généré. Pour plus d'informations concernant la manière dont les références de bibliothèque sont gérées dans le code généré, voir Chemins dans des environnement d'exécution divers 44.

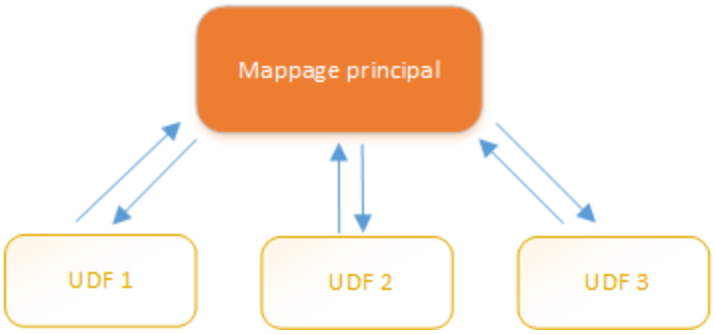
6.3 Fonctions définies par l'utilisateur

Les Fonctions définies par l'utilisateur (UDF en abrégé) sont des fonctions personnalisées définies une fois et réutilisables plusieurs fois dans le cadre du même mappage ou sur de plusieurs mappages. Les FDU sont de mini-mappages eux-mêmes : elles consistent généralement en un ou plusieurs paramètres, certains composants intermédiaires pour traiter les données et une sortie pour retourner les données à l'appelant. L'appelant est le mappage principal ou une autre UDF

Avantages des FDU

Les FDU ont les avantages suivants :

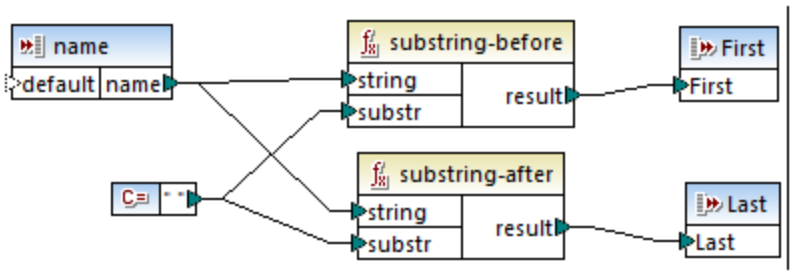
- Les FDU sont réutilisables à l'intérieur d'un mappage ou dans de multiples mappages.
- Les FDU peuvent faciliter la lecture de votre mappage : Par exemple, vous pouvez mettre en packages des parties du mappage en composants plus petits et abstraire des détails de mise en œuvre. Le diagramme ci-dessous illustre ce principe.



- Les UDF sont des fonctions flexibles qui vous permettent de procéder des strings, nombres, dates et d'autres données de manière personnalisée qui agrandit les fonctions built-in de MapForce. Par exemple, vous voulez concaténer ou séparer du texte, réaliser des calculs en avance, manipuler des dates et heures, etc.
- Une autre utilisation commune des FDU est de rechercher un champ dans un fichier XML, une base de données ou tout autre format de données pris en charge par votre édition de MapForce et présenter ces données de manière utile. Pour les détails, voir Look-up Implementation 216.
- Les FDU peuvent être appelées récursivement (par ex., la FDU s'appelle elle-même). Ceci requiert que la FDU soit définie comme fonction régulière (pas inline) 206. Les FDU récursives 214 peuvent remplir plusieurs exigences de mappage avancées, telles l'iération sur des structures de données ayant une profondeur d'enfants *N*, où *N* n'est pas connu à l'avance.

Exemple

L'exemple suivant montre une FDU simple qui partage un string en deux strings séparés. Cette FDU fait partie d'un plus grand mappage appelé **`MapForceExamples\ContactsFromPO.mfd`**. Il prend un nom en tant que paramètre (par exemple, `Helen Smith`), applique les fonctions intégrées **`substring-before`** et **`substring-after`**, puis retourne deux valeurs : `Helen` et `Smith`.



Dans cette section

Cette section vous explique comment travailler avec des FDU et est organisé comme suit :

- Notions fondamentales des FDU 204
- Paramètres FDU 209
- FDU récursives 214
- Implémentation de la consultation 216

6.3.1 Notions de bases des FDU

Cette rubrique explique comment créer, importer, éditer, copier-coller et supprimer les fonctions définies par l'utilisateur (abrégé UDF).

Créer des UDF

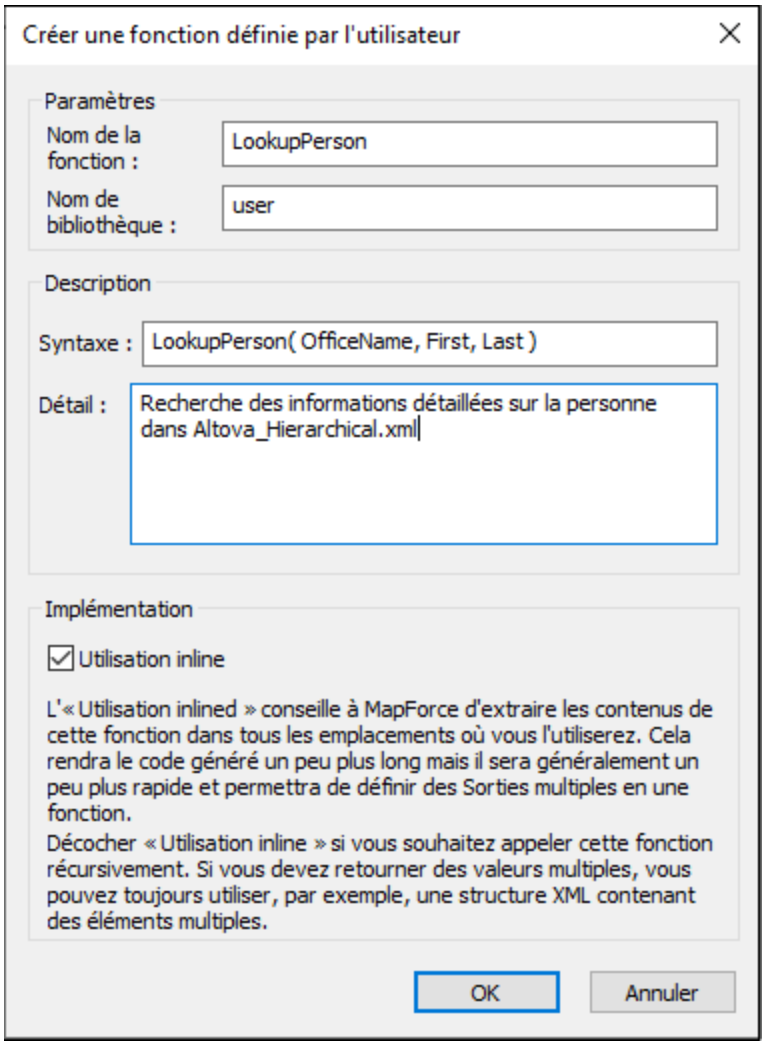
Cette sous-section explique comment créer une FDU de zéro et de composants déjà existants. L'exigence minimum est un compostant de sortie auquel quelques données sont connectées. Pour les paramètres d'entrée, une fonction peut avoir zéro, une ou plusieurs entrées. Les paramètres d'entrée et de sortie peuvent être de type simple (comme un string) ou de type complexe (une structure). Pour plus d'information concernant les paramètres simples et complexes, voir Paramètres UDF 209.

UDF à partir de zéro

Pour créer une FDU à partir de zéro, suivez les instructions ci-dessous :

1. Sélectionnez **Fonction | Créer des fonctions définies par l'utilisateur**. En alternative, cliquez sur le bouton de la barre d'outils .

2. Saisissez l'information pertinente dans le dialogue **Créer une fonction définie par l'utilisateur** (*voir la capture d'écran ci-dessous*).



Les options disponibles sont recensées ci-dessous :

- *Nom de la fonction :* Champ obligatoire. Pour le nom de votre UDF, vous pouvez utiliser le caractère suivant : caractères alphanumériques (a-z, A-Z, 09), un trait de soulignement (_), un trait d'union/tiret (-), et un deux-points (:).
- *Nom de la bibliothèque :* Champ obligatoire. Il s'agit du nom de la bibliothèque de fonction (dans la fenêtre Bibliothèques 21) dans laquelle votre fonction sera enregistrée. Si vous ne spécifiez pas le nom de bibliothèque, la fonction sera placée dans une bibliothèque par défaut appelée `user`.
- *Syntaxe :* Champ optionnel. Saisir un texte décrivant brièvement la syntaxe de la fonction (par ex., paramètres attendus). Ce texte sera affiché à côté de la fonction dans la fenêtre **Bibliothèques** et n'influera pas sur la mise en place de la fonction.
- *Détail :* Champ optionnel. Cette description sera affichée lorsque vous déplacez le curseur sur la fonction dans la fenêtre **Bibliothèques** ou dans d'autres contextes.
- *Utilisation Inline :* Sélectionnez cette case à cocher si la fonction doit être créée comme inline. Pour plus d'informations, voir *UDF régulières vs. Inline* ci-dessous.

3. Cliquez sur **OK**. La fonction devient immédiatement visible dans la fenêtre **Bibliothèques** sous le nom de bibliothèque spécifié ci-dessus. De même, la fenêtre de mappage est maintenant redessinée afin de

vous permettre de créer une nouvelle fonction (il s'agit d'un mappage autonome dont la référence est le *mappage de la fonction*). Le mappage de la fonction inclut un composant de sortie par défaut.

4. Ajoutez tous les composants requis du mappage de la fonction. Vous pouvez faire ceci comme pour un mappage standard.

Pour utiliser l'UDF dans un mappage, glissez-la depuis la fenêtre **Bibliothèques** vers la zone principale de mappage. Voir aussi *Appeler et importer les UDF* ci-dessous.

UDF de composants existants
Pour créer une UDF depuis des composantes existantes, suivez les étapes suivantes :

1. Choisir les composants pertinents sur le mappage en effectuant une sélection rectangulaire avec la souris. Vous pouvez aussi choisir plusieurs composants en cliquant sur chacun d'entre eux tout en maintenant la touche **Ctrl** appuyée.
2. Sélectionnez la commande de menu **Fonction | Créer une fonction définie par l'utilisateur depuis la sélection**. En alternative, cliquez sur le bouton de la barre d'outils  .
3. Suivez les étapes 2 à 4 depuis l'*UDF à partir de zéro*.

UDF régulières vs. Inline
Il existe deux types de FDU : *regular* et *inline*. Vous pouvez spécifier le type de votre FDU quand vous le créez. Les fonctions inline et regular se comportent différemment en termes de génération de code, de récursivité et la capacité d'avoir plusieurs paramètres de sortie. La table ci-dessous résume les différences principales entre les FDU regular et inline.

Fonctions Inline (bordure en pointillés)	Fonctions régulières (bordure solide)
Avec les fonctions inline, le code FDU est inséré dans tous les emplacements là où la fonction est appelée. Si la FDU est appelée plusieurs fois, le code de programme généré serait plus long de manière significative.	Le code de la FDU est généré une fois et les entrées sont passées comme valeurs de paramètre. Si la FDU est appelée plusieurs fois, la FDU est évaluée chaque fois avec les valeurs de paramètre correspondantes.
Les fonctions inline peuvent avoir plusieurs sorties et donc retourner plusieurs valeurs.	Les fonctions régulières ne peuvent avoir qu'une sortie. Pour retourner de multiples valeurs, vous pouvez déclarer la sortie comme étant de type complexe (par exemple, structure XML), qui vous permettrait de passer de multiples valeurs à l'appelant.
Les fonctions inline ne peuvent pas être appelées récursivement.	Les fonctions régulières peuvent être appelées récursivement.
Les fonctions inline ne prennent pas en charge la configuration d'un contexte de priorité 416 dans un paramètre.	Les fonctions régulières prennent en charge la configuration un contexte de priorité dans un paramètre.

Note : Basculer un formulaire FDU de inline à régulier, et vice versa, peut affecter le contexte de mappage 407, et ceci pourrait amener le mappage à produire un résultat différent.

Appeler et importer les UDF

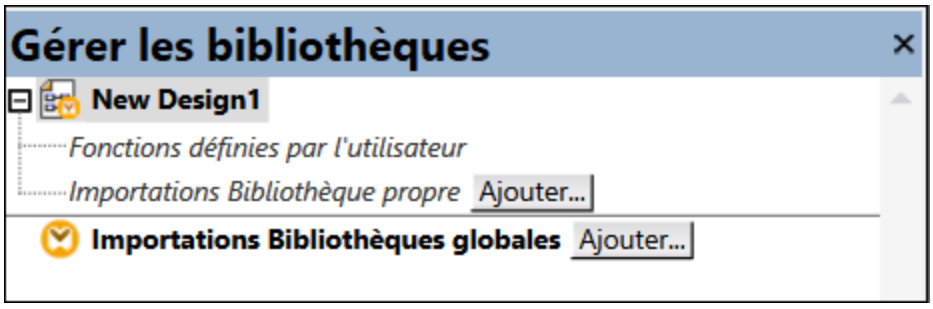
Après avoir créé une FDU, vous pouvez l'appeler du même mappage dans lequel vous l'avez créé ou depuis tout autre mappage.

Appeler la FDU du même mappage
Pour appeler une FDU depuis le même mappage, suivez les étapes suivantes ci-dessous :

1. Trouver la fonction pertinente dans la fenêtre **Bibliothèques** sous la bibliothèque que vous avez spécifiée quand vous avez créé la fonction. Pour ce faire, commencez à taper le nom dans la fenêtre **Bibliothèques**.
2. Glissez la fonction depuis la fenêtre **Bibliothèques** dans le mappage. Vous pouvez désormais connecter tous les paramètres requis à la fonction.

Importer la FDU d'un mappage différent
Pour importer une FDU à partir d'un autre mappage, suivez les instructions ci-dessous :

1. Cliquez sur la touche **Ajouter/Supprimer des bibliothèques** en bas de la fenêtre Bibliothèques 21. La fenêtre **Gérer les Bibliothèques** s'ouvre (*voir la capture d'écran ci-dessous*).



2. Pour importer des fonctions en tant que bibliothèque *locale* (uniquement dans le cadre du document actuel), cliquez sur **Ajouter** sous le nom actuel du mappage. Pour importer les fonctions en tant que bibliothèque *globale* (au niveau du programme), cliquez sur **Ajouter** à côté des **Importations Bibliothèques globales**. Lorsque vous importez une bibliothèque *localement*, vous pouvez définir le chemin du fichier de bibliothèque pour qu'il soit relatif au fichier de mappage. Avec des bibliothèques importées globalement, le chemin de la bibliothèque importée est toujours absolu.

3. Cherchez le fichier **.mfd** qui contient les fonctions, et cliquez sur **Ouvrir**. Un message apparaît vous informant qu'une nouvelle bibliothèque a été ajoutée et pouvant être accédée dans la fenêtre **Bibliothèques**.

Vous pouvez maintenant utiliser une des fonctions importées dans le mappage actuel en les glissant depuis la fenêtre **Libraries** dans le mappage. Pour plus informations concernant la consultation et l'organisation des bibliothèques de fonction, voir Gérer des bibliothèques de fonction 198.

Mappage avec identifiants (Enterprise Edition)
Si le fichier **.mfd** importé contient des identifiants, ceux-ci sont affichés en tant qu'importés (avec un arrière-plan jaune) dans le Gestionnaire d'identifiants. Par défaut, les identifiants importés ne sont pas enregistrés avec le mappage principal, mais vous pouvez créer en option une copie locale et les enregistrer sous le mappage principal.

Éditer les UDF

Pour éditer une FDU, suivez les étapes ci-dessous :

1. Ouvrir le mappage qui contient une FDU.
2. Double-cliquez sur la barre de titre de la FDU dans le mappage pour voir les contenus de la fonction là où vous pouvez ajouter, éditer ou supprimer les composants, tel que requis.
3. Pour changer les propriétés de la fonction (par ex., le nom de la description), cliquez avec la touche de droite sur une zone vide dans le mappage et sélectionnez **Paramètres de fonction** depuis le menu contextuel. En alternative, cliquez sur la touche de la barre d'outils .

Vous pouvez aussi éditer une fonction en double-cliquant sur son nom dans la fenêtre **Bibliothèques**. Néanmoins, seules des fonctions se trouvant dans le document actif actuellement peuvent être ouvertes de cette manière. Double-cliquer sur une fonction définie par l'utilisateur qui a été créée dans un autre mappage ouvre ce mappage dans une autre fenêtre. Si vous éditez ou supprimez une fonction définie par l'utilisateur qui a été importée dans plusieurs mappages, tous ces mappages seront touchés par la modification.

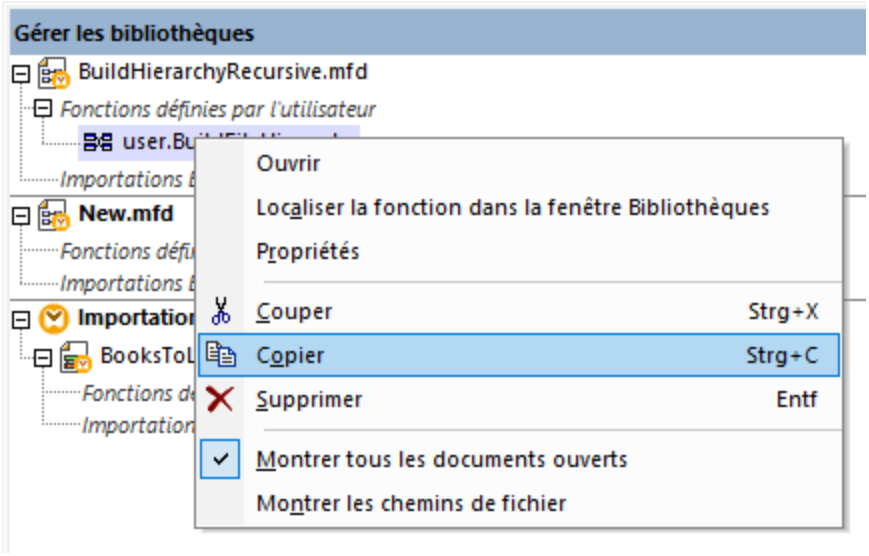
Pour retourner au mappage principal, cliquer sur la touche dans le coin supérieur gauche de la fenêtre de mappage.

MapForce vous permet de naviguer à travers différents mappages et FDU en utilisant les touches de la barre d'outils et . Les raccourcis de clavier correspondants pour ces touches sont **Alt+Left** et **Alt+Right**, respectivement.

Copier-coller les UDF

Pour copier une FDU à le collez dans un d'un autre mappage, suivez les étapes ci-dessous :

1. Ouvre la fenêtre de gestion des bibliothèques 198.
2. Cliquez avec la touche de droite dans une zone vide de la fenêtre **Bibliothèques** et choisissez l'option **Afficher tous les documents ouverts**.
3. Ouvrir les mappages de source et destination. Assurez-vous que les deux mappages sont enregistrés sur le disque. Ceci assure une résolution correcte des chemins. Voir aussi Copier-coller et les chemins relatifs 42.
4. Cliquez avec la touche de droite sur la FDU pertinente depuis le mappage de source dans la fenêtre **Gérer les bibliohèques** et sélectionnez **Copier** depuis le menu contextuel (*voir la capture d'écran ci-dessous*) ou appuyez sur **Ctrl+C**. Laissez la fenêtre **Gérer les bibliothèques** ouverte.



5. Basculer à la fenêtre de de mappage de destination (et la fenêtre **Gérer les bibliothèques** sera modifiée conformément), cliquez avec la touche de droite sur *Fonctions définies par l'utilisateur*, et sélectionnez **Coller** depuis le menu contextuel.

Supprimer les UDF

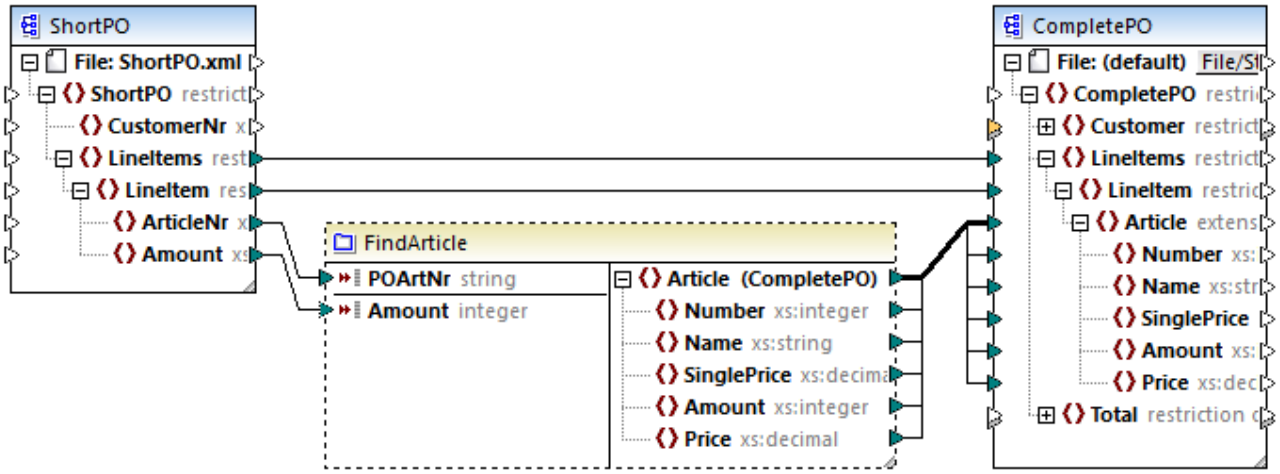
Pour supprimer une FDU, suivez les étapes ci-dessous :

1. Double-cliquer sur la barre de titre de la FDU dans le mappage.
2. Cliquer sur la touche  dans le coin supérieur droit de la fenêtre Mappage.
3. Si la fonction est utilisée dans le mappage ouvert actuel, MapForce demandera si vous voulez remplacer toutes les instances avec des composants internes. Cliquer sur **Oui** si vous souhaitez supprimer la fonction et remplacer toutes les instances dans lesquelles elle est appelée avec les composants de la fonction. Cela vous permet de garder valide le mappage principal même si la fonction est supprimée. Néanmoins, si la fonction supprimée est utilisée dans d'autres mappages externes, ceux-ci ne seront pas valides. Cliquez sur **Non** si vous voulez supprimer la fonction et tous ses composants internes en permanence. Dans ce cas, tous les mappages où la fonction est utilisée ne sera pas valide.

6.3.2 Paramètres UDF

Lorsque vous créez une FDU, vous devez spécifier quels paramètres d'entrée celle-ci doit prendre (le cas échéant) et quelle sortie elle doit retourner. Alors que des paramètres d'entrée ne sont pas toujours nécessaires, un paramètre de sortie est obligatoire dans tous les cas, c'est-à-dire qu'une fonction doit toujours retourner quelque chose. Les paramètres de fonction peuvent être de type simple (par ex., `string` ou `integer`) ou posséder une structure complexe 210. Par exemple, la fonction définie par l'utilisateur `FindArticle` illustrée ci-dessous a deux paramètres d'entrée et un de sortie :

- `POArtNr` est un paramètre d'entrée de type simple `string`;
- `Amount` est un paramètre d'entrée de type `integer`.
- `CompletePO` est un paramètre de sortie de structure complexe XML.



Ordre du paramètre
Lorsqu'une FDU a plusieurs paramètres d'entrée ou de sortie, vous pouvez modifier l'ordre dans lequel les paramètres doivent apparaître aux appelants de cette fonction. L'ordre des paramètres dans le mappage de la fonction (en commençant par le haut) dicte l'ordre dans lequel il apparaît aux appelants de la fonction.

Important

- Les paramètres d'entrée et de sortie sont triés par leur position du haut au bas. C'est pourquoi, si vous déplacez le paramètre `input3` en haut du mappage de la fonction, il deviendra le premier paramètre de cette fonction.
- Si deux paramètres ont la même position verticale, le paramètre le plus à gauche prend la précédence.
- Dans le cas inhabituel que deux paramètres ont exactement la même position, l'ID de composant interne est utilisé automatiquement.

Structures de type complexe

Les structures sur lesquelles un paramètre dans les FDU peut être basé sont résumées dans la liste ci-dessous.

MapForce Basic Edition
- Structure de schéma XML

MapForce Professional Edition
- Structure de schéma XML
- Structure de base de données

MapForce Enterprise Edition
- Structure de schéma XML
- Structure de base de données
- Structure EDI
- Structure FlexText
- Structure de schéma JSON

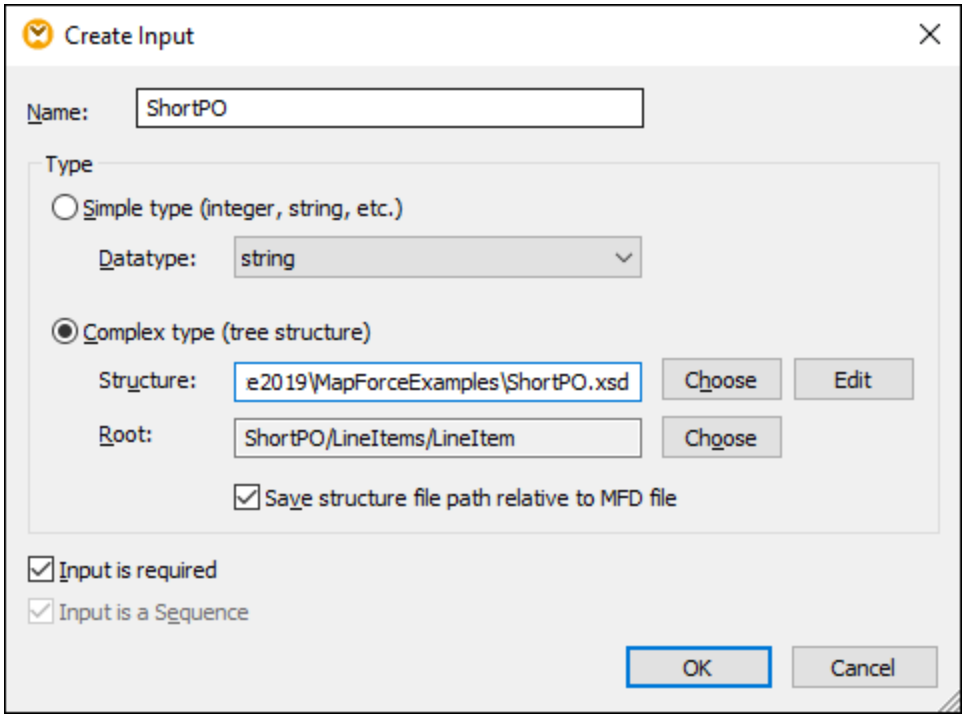
UDF basées sur les structures de base de données (éditions Professional et Enterprise)
MapForce vous permet de créer des paramètres UDF basés sur BD avec une arborescence de tables associées. L'arborescence de tables associées représente une structure in-memory qui n'a pas de connexion

vers la base de données lors de l'exécution. Ceci signifie également qu'il n'y a pas de gestion automatique de la clé étrangère et pas d'actions de table dans les paramètres ou variables.

Ajouter des paramètres

Pour ajouter un paramètre d'entrée ou de sortie, suivez les étapes suivantes :

1. Créer une FDU[204] ou ouvrir une tâche existante[207].
2. Exécuter la commande de menu **Fonction | Insérer entrée**, ou **Fonction | Insérer sortie**, respectivement (*voir la capture d'écran ci-dessous*). De manière alternative, cliquez sur (**Insérer entrée**) ou (**Insérer sortie**) dans la barre d'outils.



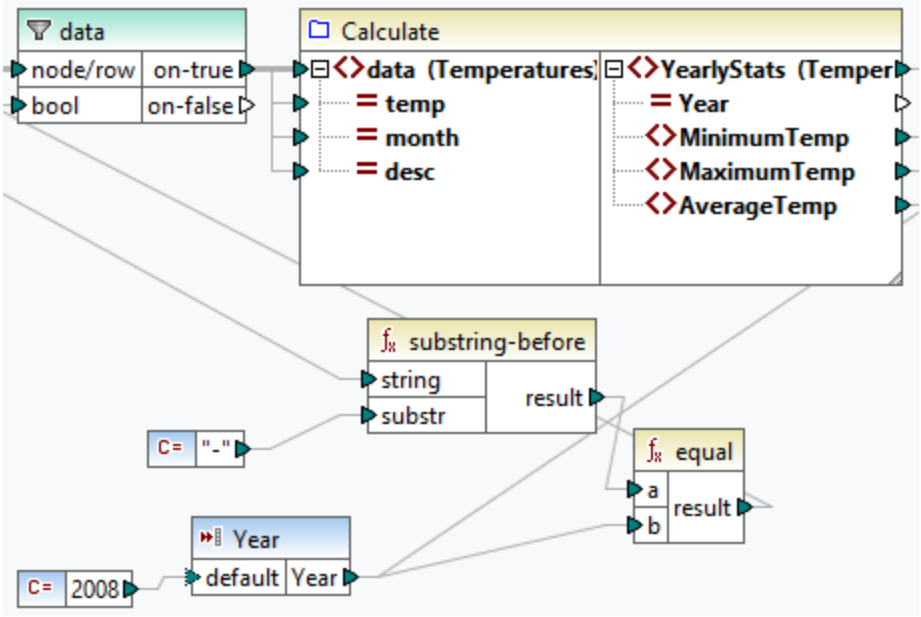
3. Choisissez si les paramètres d'entrée ou de sortie doivent être de type simple ou complexe (*voir la boite de dialogue ci-dessus*). Voir la liste des structures complexes disponibles ci-dessous. Par exemple, pour créer un paramètre qui est de type XML complexe, cliquez sur **Choisir** à côté de *Structure* et cherchez le schéma XML qui décrit la structure requise.

Si le mappage de la fonction contient déjà des schémas XML, ils sont disponibles pour la sélection en tant que structures. Sinon, vous pouvez sélectionner un nouveau schéma qui fournira la structure du paramètre. La même chose vaut pour les bases de données ou autres structures complexes si elles sont prises en charge par votre édition de MapForce. Avec des structures XML, il est possible de sélectionner un élément root pour votre structure, si le schéma XML le permet. Afin de spécifier un élément root, cliquez sur **Choisir** à côté de *Root* et sélectionner l'élément root à partir duquel le dialogue qui s'ouvre.

Une fois cochée, la case *Enregistrer le chemin de fichier de la structure relatif au fichier MFD* modifiera le chemin absolu du fichier de structure en un chemin relatif au mappage actuel, lorsque vous enregistrez le mappage. Pour plus d'informations, voir Chemins relatifs et absolus[41]. Les cases à cocher *Entrée requise* et *Entrée est une séquence* sont expliquées dans les sous-sections suivantes.

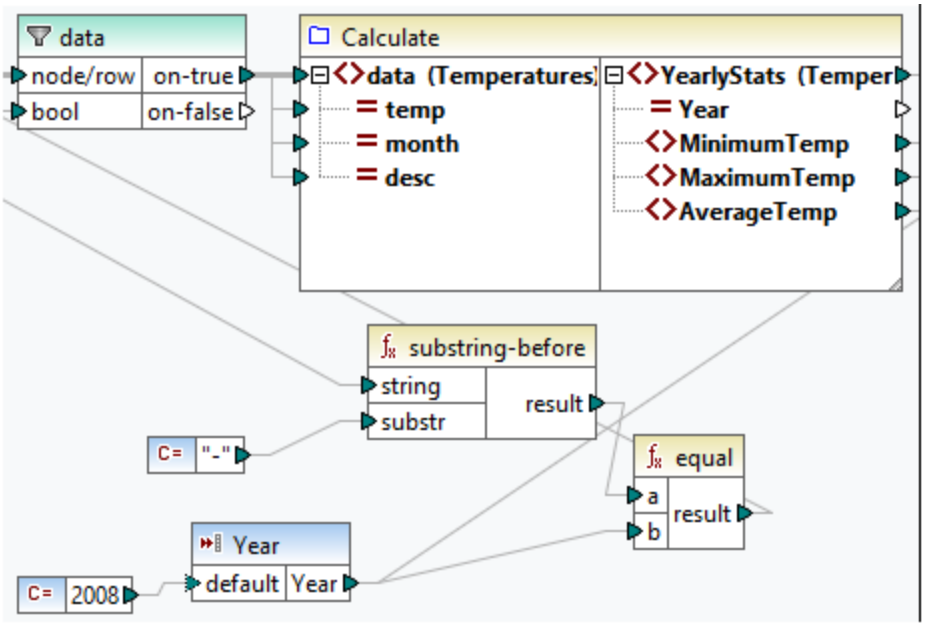
L'entrée est requise
Pour rendre un paramètre obligatoire dans une FDU, sélectionnez la case *Entrée est requise* (*voir la boîte de dialogue ci-dessus*). Si vous effacez la case à cocher *Entrée requise*, le paramètre deviendra optionnel et aura une bordure en pointillés dans le mappage.

Vous pouvez aussi spécifier une valeur de paramètre par défaut en la connectant à l'entrée par `default` d'un paramètre (*voir l'exemple ci-dessous*). La valeur par défaut sera uniquement appliquée s'il n'y a pas d'autre valeur. Si le paramètre optionnel reçoit une valeur lorsque la fonction est appelée, alors cette valeur prend le dessus sur le défaut.

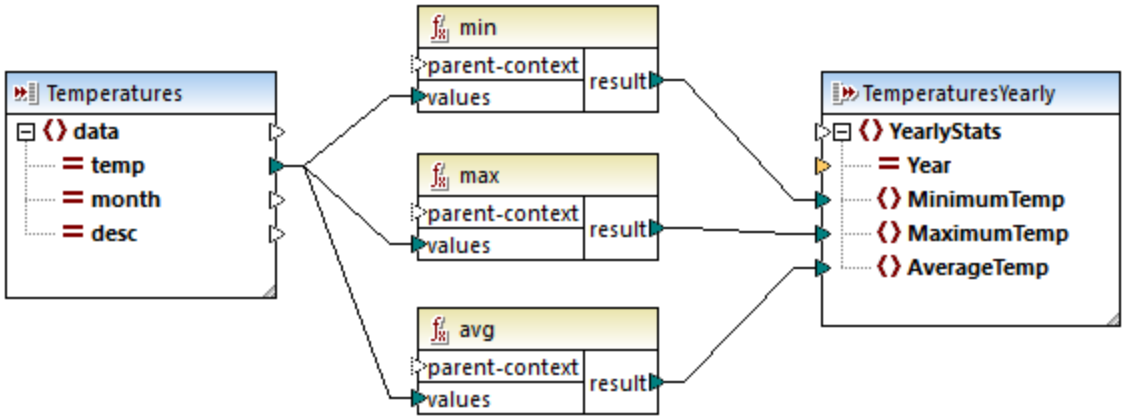


L'entrée est une séquence
En option, vous pouvez définir si un paramètre de fonction doit être traité en tant que valeur simple (option par défaut) ou en tant que séquence. Une séquence est une plage de zéro ou de plusieurs valeurs. Une séquence peut être utile quand votre fonction définie par l'utilisateur s'attend à des données d'entrée en tant que séquence afin de calculer les valeurs dans cette séquence, par exemple, en appelant les fonctions telles que **`avg`**, **`min`**, **`max`**. Pour traiter l'entrée du paramètre en tant que séquence, sélectionnez la case à cocher *Entrée est une séquence*. Veuillez noter que cette boîte à cocher est activée uniquement si la FDU est regular 206.

L'usage d'une séquence est illustrée dans le mappage suivant : **`MapForceExamples\InputIsSequence.mfd`**. Dans l'extrait de ce mappage (*voir la capture d'écran ci-dessous*), le filtre des `données` est connecté à la FDU appelée `Calculer`. La sortie du filtre est une séquence d'items. C'est la raison pour laquelle le paramètre d'entrée de la fonction est défini en tant que séquence.



Vous trouverez ci-dessous l'illustration de la fonction `Calculer` qui agrège toutes les valeurs de séquence : Elle exécute toutes les fonctions **`avg`**, **`min`**, et **`max`** dans la séquence d'entrée. Pour voir la structure interne de la fonction `Calculer`, double-cliquez sur l'en-tête du composant `Calculer` dans le mappage ci-dessus.



En règle générale, les données d'entrée, soit séquence, (séquence ou non séquence) détermine combien de fois la fonction est appelée :

- Lorsque les données d'entrée sont connectées à un paramètre *séquence*, la fonction définie par l'utilisateur est appelée uniquement *une seule fois* et la séquence complète est passée dans la fonction définie par l'utilisateur.
- Lorsque les données d'entrée sont connectées à un paramètre *non-sequence*, la fonction définie par l'utilisateur est appelée *une seule fois dans chaque item unique dans la séquence*.
- Si vous connectez une séquence vide dans un paramètre de non-séquence la fonction ne sera pas appelée du tout. Cela peut se produire si la structure de source dispose d'items optionnels ou si une condition de filtre ne retourne pas d'items correspondants. Pour éviter ceci, utilisez la fonction

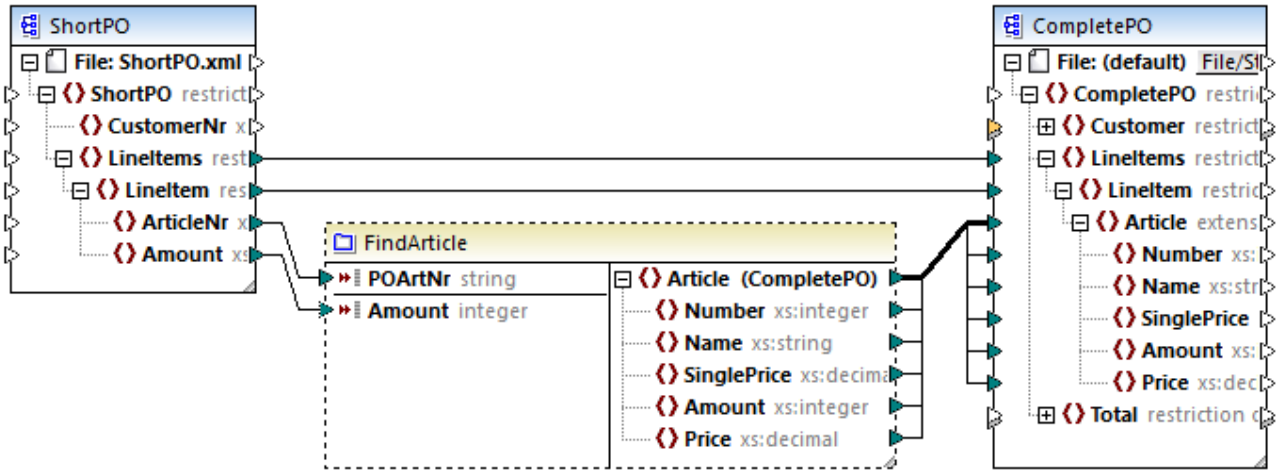
`substitute-missing` 304 avant l'entrée de fonction pour assurer que la séquence n'est jamais vide. En alternative, spécifiez le paramètre comme a séquence et ajoutez la gestion pour la séquence vide à l'intérieur de la fonction.

La case à cocher *Sortie est une séquence* peut aussi être nécessaire pour des paramètres de sortie. Lorsqu'une fonction passe une séquence de plusieurs valeurs dans son composant de sortie, et que le composant de sortie n'est pas défini sur séquence, la fonction ne retournera que le premier item dans la séquence.

6.3.3 FDU récursives

Cette rubrique explique comment rechercher des données dans le fichier XML source avec l'aide d'une FDU récursive. Pour tester la FDU récursive, vous aurez besoin du mappage suivant : **`MapForceExamples\RecursiveDirectoryFilter.mfd`**. Dans le mappage ci-dessous, la FDU `FilterDirectory` reçoit des données depuis le fichier source `Directory.xml` et le composant d'entrée simple `SearchFor` qui fournit l'extension `.xml`. Une fois que les données ont été traitées par la FDU, elles sont mappées vers le fichier cible.

Le mappage principal (*voir la capture d'écran ci-dessous*) décrit la mise en page du mappage général. La manière dont la FDU traite les données est définie séparément dans le mappage de fonction (*voir l'implémentation FDU ci-dessous*).



Objectifs
Notre objectif est de recenser des fichiers avec une extension **`.xml`** dans la sortie tout en préservant toute la structure du répertoire. Les sous-sections ci-dessous expliquent le processus de mappage en détail.

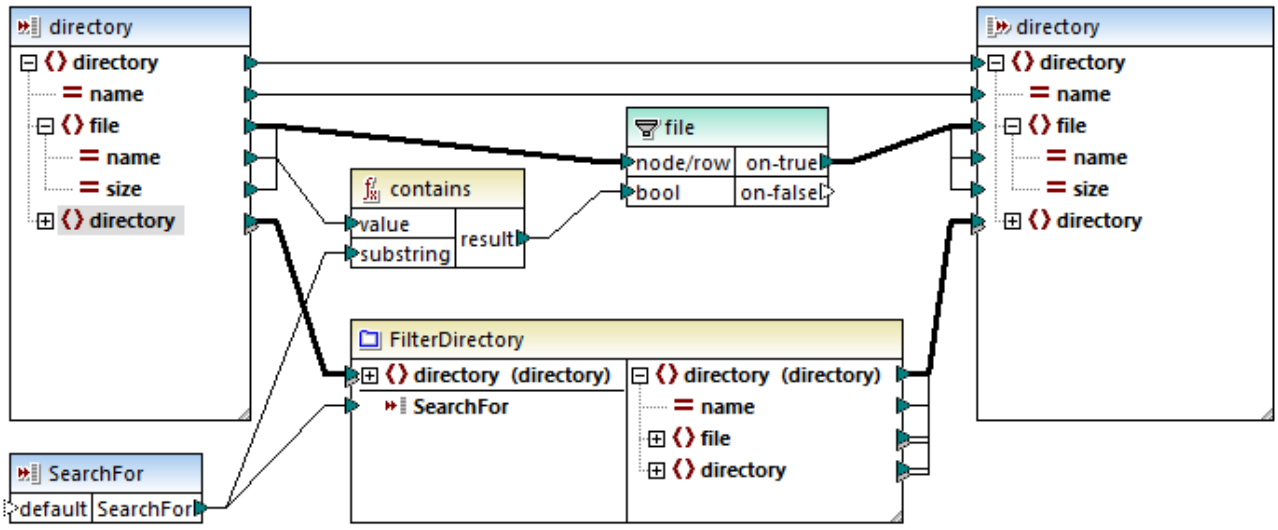
Fichier source

Ci-dessous, vous trouverez un extrait du fichier XML source (**`Directory.xml`**) qui contient des informations de fichiers et répertoires. Notez que les fichiers dans la liste ont différentes extensions (par ex., **`.xml`**, **`.dtd`**, **`.sps`**). Conformément au schéma (**`Directory.xsd`**), l'élément `directory` peut avoir des enfants `fichier` et des enfants `répertoire`. Tous les éléments `directory` sont récursifs.


```
<directory name="ExampleSite">
  <file name="blocks.sps" size="7473"/>
  <file name="blocks.xml" size="670"/>
  <file name="block_file.xml" size="992"/>
  <file name="block_schema.xml" size="1170"/>
  <file name="contact.xml" size="453"/>
  <file name="dictionaries.xml" size="206"/>
  <file name="examplesite.dtd" size="230"/>
  <file name="examplesite.spp" size="1270"/>
  <file name="examplesite.sps" size="20968"/>
  ...
  <directory name="output">
     <file name="examplesite1.css" size="3174"/>
     <directory name="images">
        <file name="blank.gif" size="88"/>
        <file name="block_file.gif" size="13179"/>
        <file name="block_schema.gif" size="9211"/>
        <file name="nav_file.gif" size="60868"/>
        <file name="nav_schema.gif" size="6002"/>
     </directory>
  </directory>
</directory>
```

Implémentation FDU

Pour voir l'implémentation interne de la FDU, double-cliquez sur son en-tête dans le mappage principal. La FDU est récursive, c'est-à-dire qu'elle contient un appel vers elle-même. Étant donné qu'elle est connectée à l'élément récursif `directory`, cette fonction sera appelée autant de fois qu'il existe des éléments `directory` imbriqués dans l'instance XML de source. Pour prendre en charge des appels récursifs, la fonction doit être regular[206].



L'implémentation de la FDU est constituée de deux parties : (i) définir les fichiers et (ii) définir le répertoire à rechercher.

Définir des fichiers
La FDU traite les fichiers comme suit : La fonction **`contains`** vérifie si le premier string (le nom de fichier) contient le sous-string `.xml` (fourni par le composant d'entrée simple `SearchFor`). Si la fonction retourne `true`, le nom de fichier avec une extension `.xml` est écrite dans la sortie.

Traiter des répertoires enfant
Les répertoires enfant du répertoire actuel sont envoyés comme entrée dans la FDU actuelle. La FDU itère donc à travers tous les éléments du `directory` et vérifie si les fichiers avec l'extension **`.xml`** existent.

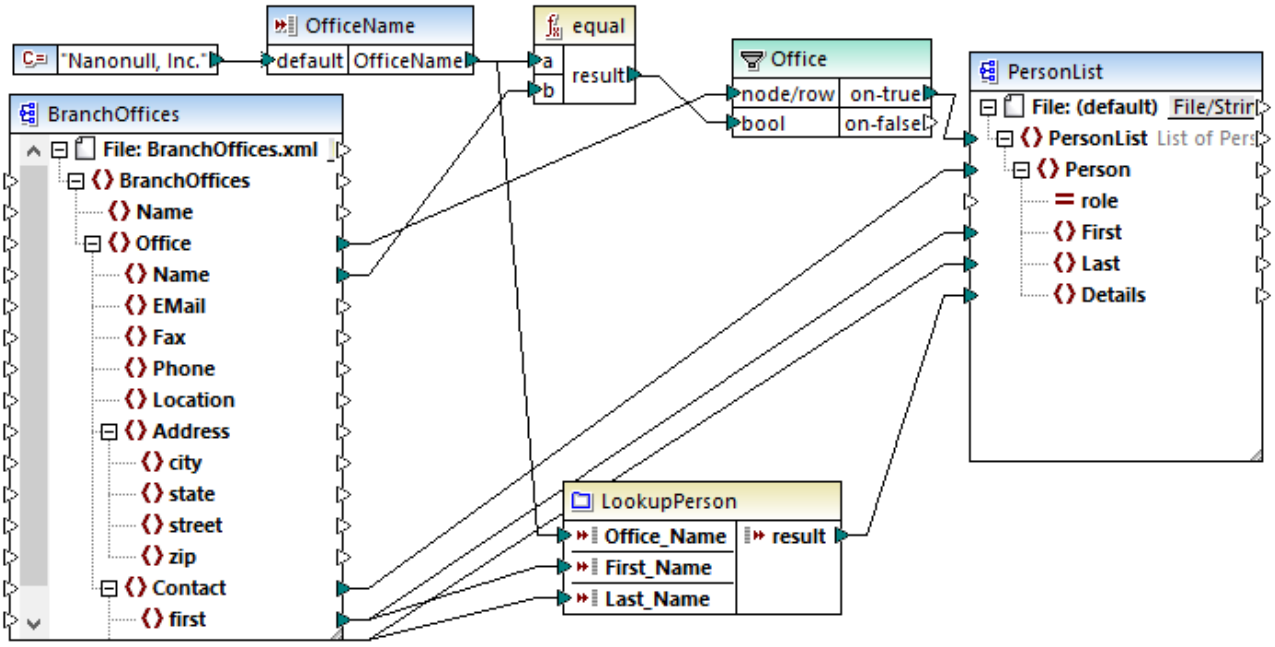
Sortie

Lorsque vous cliquez sur le volet **Sortie**, MapForce affichera uniquement des fichiers avec l'extension **`.xml`** (*voir l'extrait ci-dessous*).

```
<directory name="ExampleSite">
  <file name="blocks.xml" size="670"/>
  <file name="block_file.xml" size="992"/>
  <file name="block_schema.xml" size="1170"/>
  <file name="contact.xml" size="453"/>
  ...
  <directory name="output">
     <directory name="images"/>
  </directory>
</directory>
```

6.3.4 Implémentation de la consultation

Cette rubrique explique comment rechercher des données sur les employés et présenter cette information de manière utile. Pour tester l'implémentation de la consultation, vous aurez besoin du mappage suivant : **`MapForceExamples\PersonListByBranchOffice.mfd`**.



Objectifs
Nos objectifs sont les suivants :

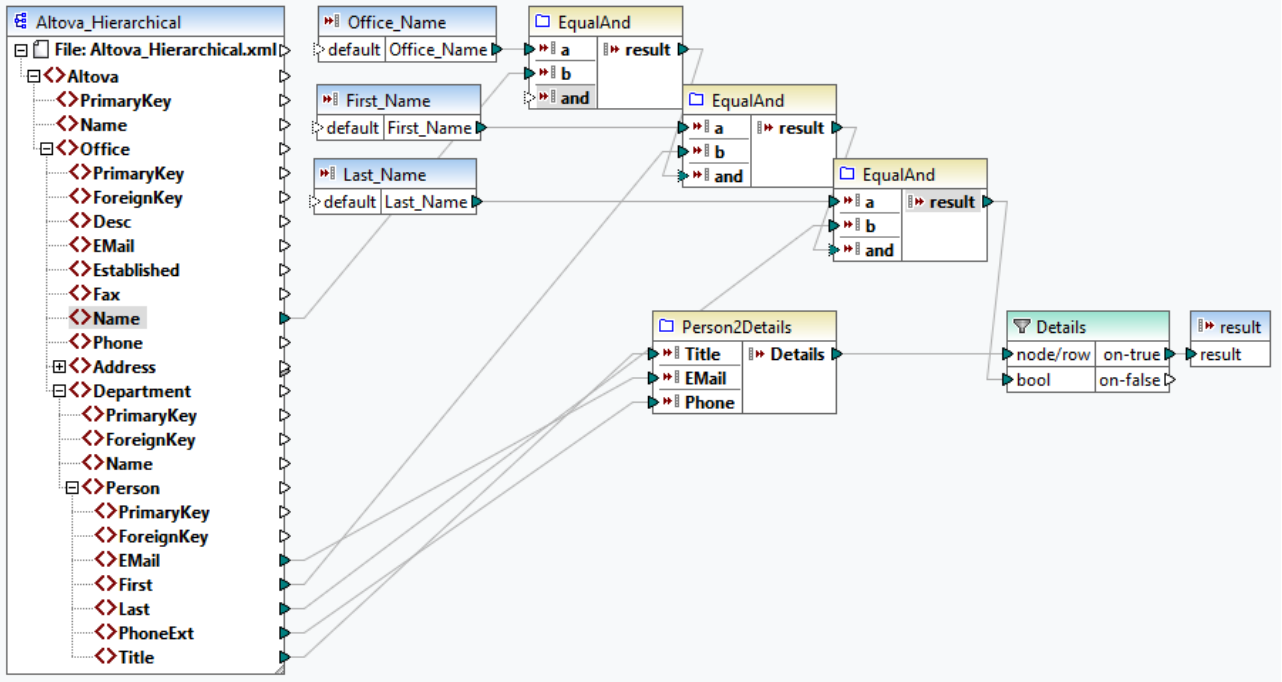
- Pour rechercher des données sur chaque employé (leur poste, adresse e-mail et titre) dans un fichier XML.
- Pour présenter ces données en tant que liste séparée par une virgule et mapper cette liste dans l'élément `Details` de l'XML cible.
- Pour extraire l'information sur les employés uniquement d'un branch office dénommé Nanonull, Inc.

Pour atteindre ces objectifs, nous avons conçu notre mappage de la manière suivante :

- Pour filtrer uniquement des employés de Nanonull, Inc., le mappage utilise le filtre `Office`.
- Pour consulter des informations sur les employés dans un autre fichier XML, le mappage appelle la FDU `LookupPerson`. L'implémentation de cette UDF est décrite dans la sous-section ci-dessous.
- Pour traiter des données d'employés, la fonction `LookupPerson` appelle en interne d'autres fonctions qui extraient et concatène l'information sur chaque employé. Toutes ces opérations se trouvent dans le mappage de la fonction et ne sont pas visibles dans le mappage principal. La fonction `LookupPerson` mappe ensuite les données de l'employé vers l'élément `Details` dans `PersonList`.

Implémentation de consultation

La fonction de consultation est fournie par la fonction `LookupPerson`, dont la définition est illustrée ci-dessous. Pour voir l'implémentation interne de la FDU, double-cliquez sur son en-tête dans le mappage principal.

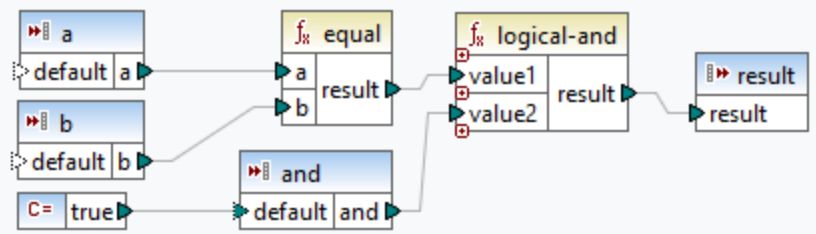


La FDU est définie comme suit :

- Les données sont extraites depuis le fichier XML **`Altova_Hierarchical.xml`**: (i) le nom de l'office et les prénom et nom de famille des employés, qui sont utilisés pour sélectionner les employés de Nanonull, Inc., et (ii) l'e-email, le titre et l'extension téléphonique qui sont concaténés en un string. Les définitions des fonctions `EqualAnd` et `Person2Detail` sont décrites ci-dessous.
- L'UDF a trois paramètres d'entrée qui fournissent les valeurs look-up `Office_Name`, `First_Name` et `Last_Name`. La valeur du paramètre `Office_Name` est extraite depuis l'entrée `OfficeName` du mappage principal, et les valeurs de `First_Name` et `Last_Name` sont fournies par le composant `BranchOffices` du mappage principal.
- La valeur de la fonction `EqaulAnd` (`true` ou `false`) est passée au filtre `Details` à chaque fois que de nouveaux détails liés aux employés (titre, e-mail, téléphone) sont traités. Lorsque le filtre `Détails` obtient la valeur `true`, l'opération look-up a du succès et les détails de l'employé sont extraits et retournés dans le mappage principal. Sinon, l'item suivant est examiné dans le contexte, et cette procédure continue jusqu'à la fin de la boucle.

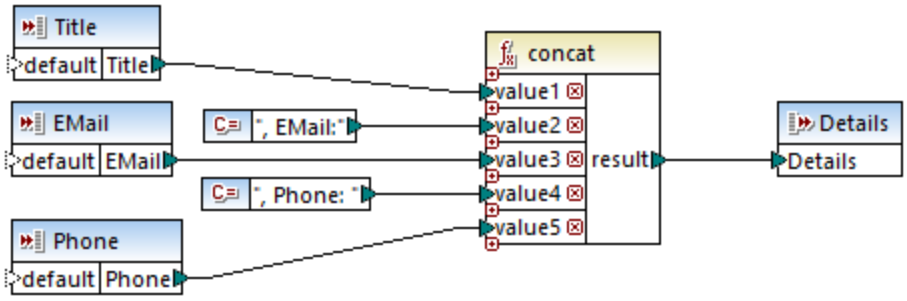
Implémentation EqualAnd

La fonction `EqualAnd` est une FDU séparée (*voir ci-dessous*) définie à l'intérieur de la FDU `LookupPerson`. Pour voir la structure interne de l'UDF `EqualAnd`, double-cliquez sur l'en-tête de la fonction.



La FDU `EqualAnd` vérifie d'abord si `a` est égale à `b` ; si le résultat est `true`, il est passé comme premier paramètre de la fonction **`logical-and`**. Si les deux valeurs dans la fonction **`logical-and`** sont true, le résultat est également true et passé à la prochaine fonction `EqualAnd`. Le résultat de la troisième fonction `EqaulAnd` (*voir LookupPerson UDF ci-dessus*) est passé au filtre `Details`.

Implémentation Person2Detail
La FDU `Person2Details` est une autre fonction à l'intérieur de la FDU `LookupPerson`. La FDU `Person2Details` (*voir ci-dessous*) concatène trois valeurs (extraites de **`Altova_Hierarchical.xml`**) et deux constantes de texte.



Sortie

Lorsque vous cliquez sur le volet **Sortie**, MapForce afficher les prénom et nom, et les détails des employés uniquement pour Nanonull, Inc (*voir l'extrait ci-dessous*).

```
<PersonList>
  <Person>
     <First>Vernon</First>
     <Last>Callaby</Last>
     <Details>Office Manager, EMail:v.callaby@nanonull.com, Phone: 582</Details>
  </Person>
  <Person>
     <First>Frank</First>
     <Last>Further</Last>
     <Details> Accounts Receivable, EMail:f.further@nanonull.com, Phone:
471</Details>
  </Person>
  ...
</ PersonList>
```

6.4 Fonctions personnalisées

Cette section explique comment importer les bibliothèques XSLT[220].

6.4.1 Importer des fonctions XSLT personnalisées

Vous pouvez étendre les bibliothèques de fonction XSLT 1.0 et 2.0 disponibles dans MapForce avec vos propres fonctions personnalisées, si vos fonctions personnalisées retournent des types simples.

> Seules des fonctions personnalisées qui retournent des types de données simples (par exemple, strings) sont prises en charge.

Pour importer des fonctions depuis un fichier XSLT :

1. Cliquez sur la touche **Ajouter/Supprimer des bibliothèques** en bas de la fenêtre Bibliothèques[21]. La fenêtre **Gérer les Bibliothèques** s'ouvre (*voir la capture d'écran ci-dessous*).

2. Pour importer des fonctions en tant que bibliothèque *locale* (uniquement dans le cadre du document actuel), cliquez sur **Ajouter** sous le nom actuel du mappage. Pour importer les fonctions en tant que bibliothèque *globale* (au niveau du programme), cliquez sur **Ajouter** à côté des **Importations Bibliothèques globales**. Lorsque vous importez une bibliothèque *localement*, vous pouvez définir le chemin du fichier de bibliothèque pour qu'il soit relatif au fichier de mappage. Avec des bibliothèques importées globalement, le chemin de la bibliothèque importée est toujours absolu.

Chercher le fichier .xsl qui contient les fonctions, et cliquer sur **Ouvrir**. Un message apparaît vous informant qu'une nouvelle bibliothèque a été ajoutée.

Les fichiers XSLT importés apparaissent sous forme de bibliothèques dans la fenêtre Bibliothèques, et affichent tous les modèles nommés en tant que fonctions sous le nom de bibliothèque. Si vous ne voyez pas la bibliothèque importée, veuillez vous assurer que vous avez bien sélectionné XSLT en tant que langage de transformation[17]. Voir aussi Gérer les Bibliothèques de fonction[198].

Veuillez noter que :

- Pour pouvoir importer dans MapForce, les fonctions doivent être déclarées en tant que modèles nommés conformément aux spécifications XSLT dans le fichier XSLT. Vous pouvez aussi importer des fonctions qui se produisent dans un document XSLT 2.0 sous la forme de `<xsl:function name="MyFunction">`. Si le fichier XSLT importé ou comprend d'autres fichiers XSLT, alors ces fichiers et fonctions XSLT seront également importés.
- Les connecteurs d'entrée mappables des fonctions personnalisées importées dépendent du nombre de paramètres utilisés dans l'appel du modèle ; des paramètres optionnels sont aussi pris en charge.
- Les espaces de noms sont pris en charge.
- Si vous effectuez des mises à jour dans des fichiers XSLT que vous avez déjà importé dans MapForce, des modifications seront détectées automatiquement et MapForce vous invite à recharger les fichiers.
- Lorsque vous écrivez des modèles nommés, assurez-vous que les instructions XPath utilisés dans le modèle sont liés dans les espaces de noms corrects. Pour voir les liaisons d'espace de noms du mappage, consulter le code XSLT généré 65.

Types de données dans XPath 2.0

Si votre document XML référence un schéma XML et est valide, vous devez construire explicitement ou lancer des types de données qui ne sont pas implicitement convertis dans le type de données requis par une opération.

Dans le XPath 2.0 Data Model utilisé par l'Altova XSLT 2.0 Engine, toutes les valeurs de nœud **atomisées** depuis le document XML sont attribuées au type de données `xs:untypedAtomic`. Le type `xs:untypedAtomic` fonctionne bien avec des conversions de type implicite.

Par exemple,

- l'expression `xs:untypedAtomic("1") + 1` résulte dans une valeur de 2 parce que la valeur `xdt:untypedAtomic` est **implicitement** promotée dans `xs:double` par l'opérateur d'addition.
- Les opérateurs arithmétiques promeuvent implicitement des opérandes dans `xs:double`.
- Les opérateurs de comparaison de valeur promeuvent des opérandes dans `xs:string` avant la comparaison.

Voir aussi :

Exemple : Ajouter des fonctions XSLT 1.0 personnalisées 221
Exemple : Totaliser les valeurs de nœud 224
Implémentation de moteur XSLT 1.0 486
Implémentation de moteur XSLT 2.0 487

6.4.1.1 Exemple : Ajouter des fonctions XSLT personnalisées

Cet exemple illustre comment importer des fonctions XSLT 1.0 personnalisés dans MapForce. Les fichiers nécessaires pour cet exemple sont disponibles dans le dossier suivant : **`C:\Users\<username>\Documents\Altova\MapForce2024\MapForceExamples`**.

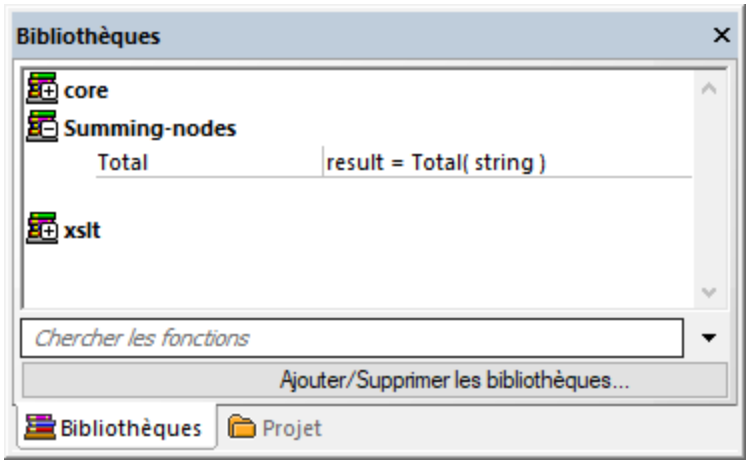
- Name-splitter.xslt. Ce fichier XSLT défini un modèle nommé appelé "**tokenize**" avec un seul paramètre "string". Le modèle fonctionne par le biais d'un string d'entrée et sépare les caractères en majuscule avec un espace pour chaque occurrence.
- Name-splitter.xml (le fichier d'instance XML de source à traiter)
- Customers.xsd (le schéma XML de source)
- CompletePO.xsd (le schéma XML de cible)

Pour ajouter une fonction XSLT personnalisée :

1. Cliquez sur la touche **Ajouter/Supprimer des bibliothèques** en bas de la fenêtre Bibliothèques[21]. La fenêtre **Gérer les Bibliothèques** s'ouvre (*voir la capture d'écran ci-dessous*).

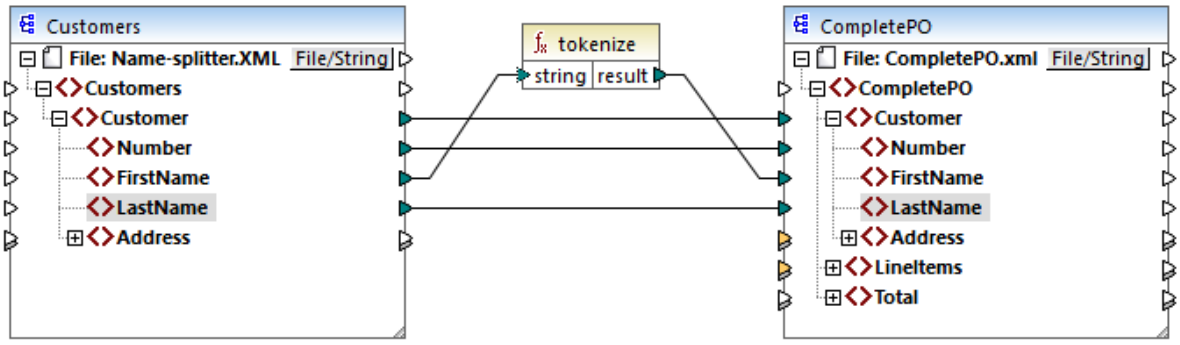
2. Pour importer des fonctions en tant que bibliothèque *locale* (uniquement dans le cadre du document actuel), cliquez sur **Ajouter** sous le nom actuel du mappage. Pour importer les fonctions en tant que bibliothèque *globale* (au niveau du programme), cliquez sur **Ajouter** à côté des **Importations Bibliothèques globales**. Lorsque vous importez une bibliothèque *localement*, vous pouvez définir le chemin du fichier de bibliothèque pour qu'il soit relatif au fichier de mappage. Avec des bibliothèques importées globalement, le chemin de la bibliothèque importée est toujours absolu.

3. Parcourez le fichier .xsl ou .xslt qui contient le modèle nommé que vous voulez utiliser pour qu'il agisse comme fonction, dans ce cas **Name-splitter.xslt**, ouis cliquez sur **Ouvrir**. Une boîte de message apparaît vous informant qu'une nouvelle bibliothèque a été ajoutée, et le nom du fichier XSLT apparaît dans la fenêtre Bibliothèques, avec les fonctions définies comme modèles nommés (dans cet exemple, **Name-splitter** avec la fonction `tokenize`).



Pour utiliser la fonction XSLT dans votre mappage :

1. Glisser la fonction `tokenize` dans la fenêtre de Mappage et mapper les items comme indiqué ci-dessous.



2. Cliquer sur l'onglet **XSLT** pour voir le code XSLT généré.

```
1   <?xml version="1.0" encoding="UTF-8"?>
2   <!-- ... -->
11  <xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:xs="http
12      <xsl:include href="file:///C:/Users/altova/Documents/Altova/MapForce2020/MapForceExamples
13      <xsl:output method="xml" encoding="UTF-8" byte-order-mark="no" indent="yes"/>
14      <xsl:template match="/">
15          <CompletePO>
16              <xsl:attribute name="xsi:noNamespaceSchemaLocation" namespace="http://www.w3.org/
    CompletePO.xsd'"/>
17              <xsl:for-each select="Customers/Customer">
18                  <Customer>
19                      <Number>
20                          <xsl:sequence select="xs:string(xs:integer(fn:string(Number)))"/>
21                      </Number>
22                      <FirstName>
23                          <xsl:call-template name="tokenize">
24                              <xsl:with-param name="string" select="FirstName" as="item()"/>
25                          </xsl:call-template>
26                      </FirstName>
27                      <LastName>
28                          <xsl:sequence select="fn:string(LastName)"/>
29                      </LastName>
30                  </Customer>
31              </xsl:for-each>
32          </CompletePO>
33      </xsl:template>
34  </xsl:stylesheet>
35
```

Note : Dès qu'un modèle nommé est utilisé dans un mappage, le fichier XSLT contenant le modèle nommé est **included** dans le code XSLT généré (**xsl:include href...**), et est **appelé** en utilisant la commande **xsl:call-template**.

3. Cliquer sur l'onglet Sortie pour voir le résultat du mappage.

Pour supprimer des bibliothèques XSLT personnalisées depuis MapForce :

1. Cliquer sur la touche **Ajouter/Supprimer Bibliothèques**, dans la zone inférieure de la fenêtre Bibliothèques. La fenêtre Gérer les Bibliothèques s'ouvre.
2. Cliquer sur **Supprimer Bibliothèque** à côté de la bibliothèque à supprimer.

6.4.1.2 Exemple : Totaliser les valeurs de nœud

Cet exemple montre comment traiter plusieurs nœud d'un document XML et faire mapper le résultat comme une valeur unique dans un document XML de cible. Plus spécifiquement, l'objectif du mappage est de calculer le prix de tous les produits dans un fichier XML de source et l'écrire en tant que valeur unique dans un fichier XML de sortie. Les fichiers utilisés dans cet exemple sont disponibles dans le dossier **<Documents>\Altova\MapForce2024\MapForceExamples\Tutorial** :

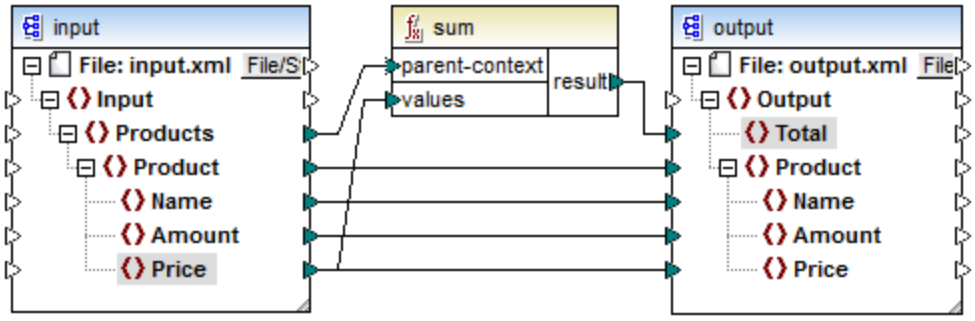
- **Summing-nodes.mfd** — le fichier de mappage
- **input.xml** — le fichier XML de source
- **input.xsd** — le schéma XML de source
- **output.xsd** — le schéma XML de cible
- **Summing-nodes.xslt** — Une feuille de style XSLT personnalisée contenant un modèle nommé pour additionner les nœuds individuels.

Il existe deux moyens différents d'atteindre l'objectif du mappage :

- En utilisant la fonction **sum** 241. Cette fonction intégrée MapForce est disponible dans la fenêtre **Bibliothèques**.
- En important une feuille de style XSLT personnalisée dans MapForce.

Solution 1: Utiliser la fonction d'agrégation "sum"

Pour utiliser la fonction **sum** dans le mappage, la glisser depuis la fenêtre **Bibliothèques** dans le mappage. Veuillez noter que les fonctions disponibles dans la fenêtre **Bibliothèques** dépendent de la version de langage XSLT que vous avez sélectionné (XSLT 1 ou XSLT 2). Ensuite, créer les connexions de mappage comme indiqué ci-dessous.



Pour plus d'informations concernant les fonctions d'agrégation de la bibliothèque **core**, voir aussi core | aggregate functions 234.

Solution 2: Utiliser une feuille de style XSLT personnalisée

Comme mentionné ci-dessus, l'objectif de l'exemple est d'additionner les champs `Price` des produits dans le fichier XML de source, dans ce cas, les produits A et 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>
```

L'extrait de code ci-dessous montre une feuille de style XSLT personnalisée qui utilise le modèle nommé "Total" et un seul paramètre `string`. Le modèle fonctionne par le biais du fichier d'entrée XML et additionne toutes les valeurs obtenues par l'expression XPath `/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>
```

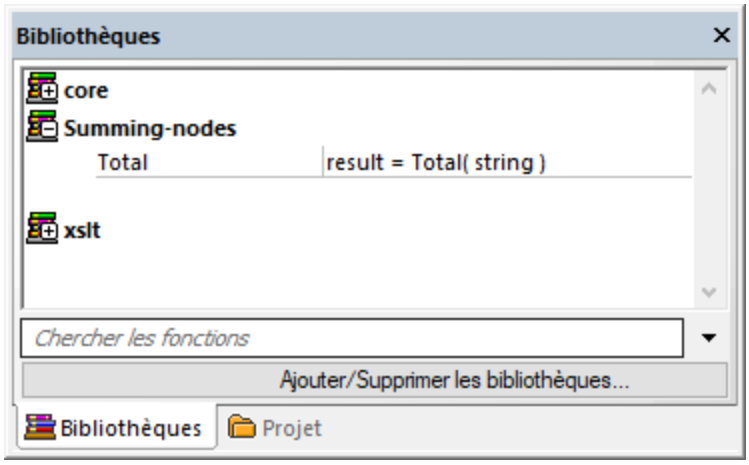
Note : Pour additionner les nœuds dans XSLT 2.0, modifier la déclaration de feuille de style en `version="2.0"`.

Avant d'importer la feuille de style XSLT dans MapForce, choisir XSLT 1.0 en tant que langage de transformation[17]. Vous êtes maintenant prêt à importer la fonction personnalisée, comme suit :

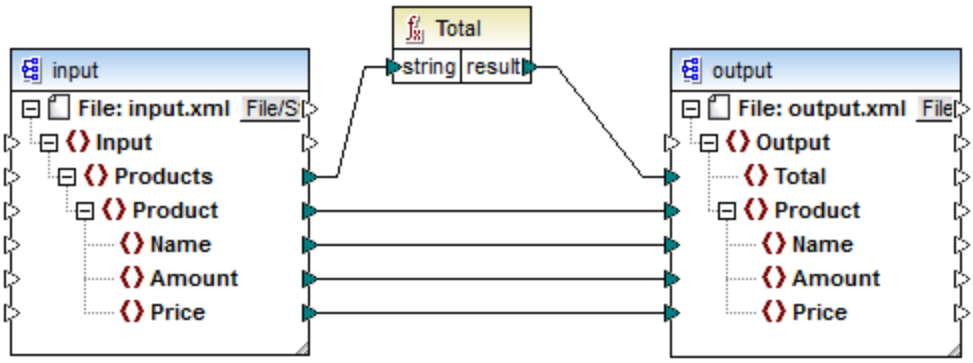
1. Cliquez sur la touche **Ajouter/Supprimer des bibliothèques** en bas de la fenêtre Bibliothèques[21]. La fenêtre **Gérer les Bibliothèques** s'ouvre (*voir la capture d'écran ci-dessous*).

2. Pour importer des fonctions en tant que bibliothèque *locale* (uniquement dans le cadre du document actuel), cliquez sur **Ajouter** sous le nom actuel du mappage. Pour importer les fonctions en tant que bibliothèque *globale* (au niveau du programme), cliquez sur **Ajouter** à côté des **Importations Bibliothèques globales**. Lorsque vous importez une bibliothèque *localement*, vous pouvez définir le chemin du fichier de bibliothèque pour qu'il soit relatif au fichier de mappage. Avec des bibliothèques importées globalement, le chemin de la bibliothèque importée est toujours absolu.

3. Chercher **<Documents>\Altova\MapForce2024\MapForceExamples\Tutorial\Summing-nodes.xslt**, et cliquer sur **Ouvrir.** Un message apparaît vous informant qu'une nouvelle bibliothèque a été ajoutée et la nouvelle bibliothèque apparaît dans la fenêtre Bibliothèques.



4. Glisser la fonction `Total` depuis les Bibliothèques dans le mappage, et créer les connexions de mappage comme indiqué ci-dessous.



Pour consulter le résultat de mappage, cliquer sur l'onglet **Sortie**. La somme de deux champs `Price` est maintenant affiché dans le champ `Total`.

```
<?xml version="1.0" encoding="UTF-8"?>
<Output xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="output.xsd">
   <Total>25</Total>
   <Product>
      <Name>ProductA</Name>
      <Amount>10</Amount>
      <Price>5</Price>
   </Product>
   <Product>
      <Name>ProductB</Name>
      <Amount>5</Amount>
      <Price>20</Price>
   </Product>
</Output>
```

6.5 Expressions régulières

Lors de la conception d'un mappage MapForce, vous pouvez utiliser des expressions régulières ("regex") dans les contextes suivants :

- Dans les paramètres **pattern** de la fonction **tokenize-regexp** 315.

La syntaxe et la sémantique d'expression régulière pour XSLT et XQuery sont tels que définis dans Annexes F de "XML Schema Part 2: Datatypes Second Edition".

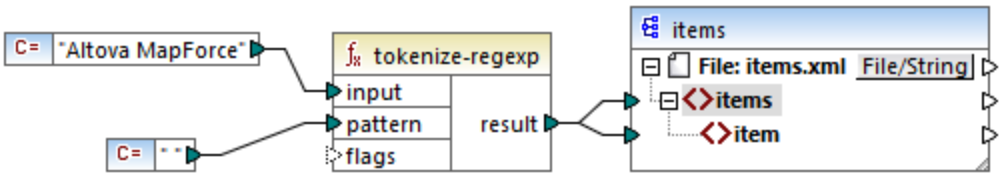
Note : Lors de la génération de code C++, C# ou Java, les fonctions avancées de la syntaxe d'expression régulière peuvent différer légèrement. Voir la documentation regex de chaque langage pour plus d'informations.

Terminologie

Examinons la terminologie d'expression régulière basique en analysant la fonction **tokenize-regexp** en tant qu'un exemple. Cette fonction partage du texte dans une séquence de strings, avec l'aide des expressions régulières. Pour ce faire, la fonction prend les paramètres d'entrée suivants :

input	Le string d'entrée à traiter par la fonction. L'expression régulière fonctionnera sur ce string.
pattern	Le pattern d'expression régulière à appliquer.
flags	Paramètre optionnel qui définit des options supplémentaires (flags) qui déterminent comment l'expression régulière est interprétée, voir "Flags" ci-dessous.

Dans le mappage ci-dessous, le string d'entrée est "Altova MapForce". Le paramètre **pattern** est un caractère d'espace et aucun flag d'expression régulière n'est utilisé.



En conséquence, le texte est partagé à chaque fois que le caractère d'espace se produit, le résultat de mappage est donc le suivant :

```
<items>
   <item>Altova</item>
   <item>MapForce</item>
</items>
```

Veuillez noter que la fonction **tokenize-regexp** exclut les caractères correspondants du résultat. Autrement dit, le caractère d'espace dans cet exemple est omis de la sortie.

L'exemple ci-dessus est très basique et le même résultat peut être obtenu sans expression régulière, avec la fonction tokenize[313]. Dans un scénario plus pratique, le paramètre **pattern** contiendrait une expression régulière plus complexe. L'expression régulière peut consister en :

- Littéraux
- Classes de caractère
- Gammes de caractère
- Classe niées
- Caractères Meta
- Quantificateurs

Littéraux

Utiliser des littéraux pour faire correspondre les caractères exactement tels qu'ils sont écrits (littéralement). Par exemple, si le string d'entrée est `abracadabra`, et **pattern** est le littéral `br`, le résultat sera :

```
<items>
   <item>a</item>
   <item>acada</item>
   <item>a</item>
</items>
```

L'explication est que le littéral `br` avait deux correspondances dans le string d'entrée `abracadabra`. Une fois avoir retiré les caractères correspondants du résultat, la séquence de trois strings illustrée ci-dessus est produite.

Classes de caractère

Si vous contenez un ensemble de caractères dans des crochets (`[` et `]`), cela crée une classe de caractère. Un seul (et uniquement un seul) des caractères contenu dans la classe de caractère a une correspondance, par exemple :

- Le pattern `[aeiou]` correspond à toute voyelle de casse minuscule.
- Le pattern `[mj]ust` correspond à "must" et "just".

Note : Le pattern est sensible à la casse, donc un "a" de casse minuscule ne correspond pas à la casse majuscule "A". Pour rendre la correspondance insensible à la casse, utiliser le flag **i**, voir ci-dessous.

Gammes de caractère

Utiliser `[a-z]` pour créer une gamme entre les deux caractères. Seul un des caractères trouvera une correspondance un à la fois. Par exemple, le pattern `[a-z]` correspond à tout caractère de casse minuscule entre "a" et "z".

Classes niées

L'utilisation du caret (`^`) en tant que le premier caractère après le crochet ouvert nie la classe de caractère. Par exemple, le pattern `[^a-z]` correspond à tout caractère ne se trouvant pas dans la classe de caractère, y compris les caractères de nouvelle ligne.

Correspondant à un caractère

Utiliser le méta-caractère du point (**.**) pour faire correspondre chaque caractère unique, sauf pour le caractère de nouvelle ligne. Par exemple, le pattern **.** correspond à tout caractère unique.

Quantificateurs

Dans le cadre d'une expression régulière, des quantificateurs définissent combien de fois le caractère ou sous-expression précédente est autorisée à se produire pour que la correspondance puisse avoir lieu.

?	Correspond à zéro ou une occurrence de l'item qui précède. Par exemple, le pattern **mo?** fera se correspondre "m" et "mo".
+	Correspond à une ou plusieurs occurrences de l'item qui précède. Par exemple, le pattern **mo+** correspondra à "mo", "moo", "mooo", etc.
*****	Correspond à zéro or plusieurs occurrences de l'item qui précède.
{min,max}	Correspond à n'importe quel nombre d'occurrences entre *min* et *max*. Par exemple, le pattern **mo{1,3}** fait se correspondre "mo", "moo" et "mooo".

Parenthèses

Les parenthèses **(** et **)** sont utilisées pour regrouper des parties d'un regex. Elles peuvent être utilisées pour appliquer des quantificateurs dans une sous-expression (contrairement à uniquement un seul caractère), ou avec une alternance (voir ci-dessous).

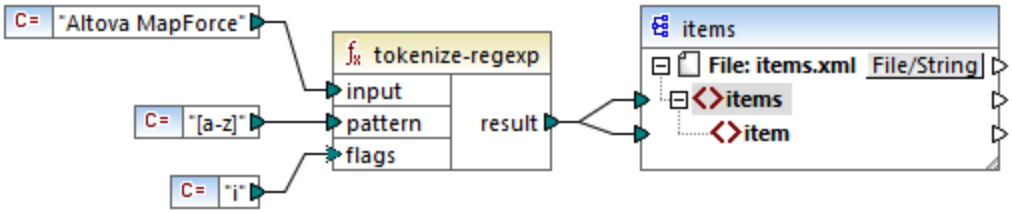
Alternance

La barre verticale **|** signifie "ou". Elle peut être utilisée pour faire correspondre n'importe laquelle des différentes sous-expressions séparées par **|**. Par exemple, le pattern **(horse|make) sense** fera se correspondre les deux "horse sense" et "make sense".

Flags

Il s'agit de paramètres optionnels qui définissent comme l'expression régulière doit être interprétée. Chaque flag correspond à une lettre. Les lettres peuvent se trouver dans n'importe quel ordre et peuvent être répétées.

s	Si ce flag est présent, le processus de correspondance fonctionne dans le mode "dot-all". Si le string d'entrée contient "hello" et "world" dans deux lignes *différentes*, l'expression régulière **hello*world** correspondra uniquement si le flag **s** est défini.
m	Si ce flag est présent, le processus de correspondance fonctionne dans le mode multi-ligne. Dans le mode multi-ligne, le symbole caret **^** correspond au début d'une ligne, c'est à dire le début du string entier et le premier caractères après un caractère de nouvelle ligne. Le caractère dollar **$** correspond à la fin d'une ligne, c'est à dire la fin du string entier et le caractère juste avant un caractère de nouvelle ligne.

	Nouvelle ligne est le caractère **#x0A**.
i	Si ce flag est présent, le processus de correspondance fonctionne dans le mode insensible à la casse. Par exemple, l'expression régulière `[a-z]` plus le flag **i** font se correspondre toutes les lettres a-z et A-Z. 
x	Si ce flag est présent, les caractères d'espace blanc sont supprimés de l'expression régulière avant le processus de mise en correspondance. Les caractères d'espace blanc sont **#x09**, **#x0A**, **#x0D** et **#x20**. **Note:** Les caractères d'espace blanc se trouvant dans une classe de caractère ne sont pas supprimés, par exemple, `[#x20]`.

6.6 Référence des bibliothèques de fonctions

Cette section de référence décrit les fonctions built-in de MapForce disponibles dans la fenêtre Bibliothèques [21]. Les fonctions sont organisées par bibliothèque. La disponibilité des bibliothèques de fonction dans la fenêtre **Bibliothèques** dépend du langage de transformation du mappage que vous choisissez pour votre mappage. Pour en savoir plus sur la liste des langages de transformation disponibles, voir cette rubrique [17].

L'information sur la compatibilité des fonctions et les langages de transformation est fournie dans les sous-sections ci-dessous.

fonctions core

La liste ci-dessous résume la comptabilité des fonctions core avec les langages de transformation.
fonctions core | aggregate

- **avg**, **max**, **max-string**, **min**, **min-string**: XSLT 2.0, XSLT 3.0, XQuery 1.0, C#, C++, Java, Built-In;
- **count**, **sum**: tous les langages de transformation.

fonctions core | conversion

- **boolean**, `string`, **number**: tous les langages de transformation ;
- **format-date**, **format-dateTime**, **format-time**: XSLT 2.0, XSLT 3.0, C#, C++, Java, Built-In;
- **format-number** : XSLT 1.0, XSLT 2.0, XSLT 3.0, C#, C++, Java, Built-In;
- **parse-date**, **parse-dateTime**, **parse-number**, **parse-time**: C#, C++, Java, Built-In.

core | file path functions
Toutes les fonctions de chemin d'accès au fichier sont compatibles avec les langages de transformation.

core | generator functions
La fonction **auto-number** est disponible pour tous les langages de transformation.

core | logical functions
les fonctions de chemin d'accès au fichier sont compatibles avec les langages de transformation.

core | math functions

- **add**, **ceiling**, **divide**, **floor**, **modulus**, **multiply**, **round**, **subtract**: tous les langages de transformation ;
- **round-precision** : C#, C++, Java, Built-In.

core | node functions

- **is-xsi-nil**, **local-name**, **static-node-annotation**, **static-node-name**: tous les langages de transformation ;
- **node-name**, **set-xsi-nil**, **substitute-missing-with-xsi-nil**: XSLT 2.0, XSLT 3.0, XQuery 1.0, C#, C++, Java, Built-In.

core | QName functions
Les fonctions `QName` sont compatibles avec les langages de transformation à l'exception de XSLT1.0.

fonctions core | sequence

- **exists**, **not-exists**, **position**, **substitute-missing**: tous les langages de transformation ;
- **distinct-values**, **first-items**, **generate-sequence**, **item-at**, **items-from-till**, **last-items**, **replicate-item**, **replicate-sequence**, **set-empty**, **skip-first-items**: XSLT 2.0, XSLT 3.0, XQuery 1.0, C#, C++, Java, Built-In;
- **group-adjacent**, **group-by**, **group-ending-with**, **group-into-blocks**, **group-starting-with**: XSLT 2.0, XSLT 3.0, C#, C++, Java, Built-In.

core | fonctions string

- **concat**, **contains**, **normalize-space**, **starts-with**, **string-length**, **substring**, **substring-after**, **substring-before**, **translate**: tous les langages de transformation ;
- **char-from-code**, **code-from-char**, **tokenize**, **tokenize-by-length**, **tokenize-regexp**: XSLT 2.0, XSLT 3.0, XQuery 1.0, C#, C++, Java, Built-In.

fonctions bson (uniquement MapForce Enterprise Edition)

Toutes les fonctions BSON sont compatibles uniquement avec Built-In.

fonctions db (MapForce Professional et Enterprise Edition)

Les fonctions `db` sont compatibles avec C#, C++, Java, Built-In.

fonctions edifact (uniquement MapForce Enterprise Edition)

Les fonctions `edifact` sont compatibles avec C#, C++, Java, Built-In.

fonctions lang (MapForce Professional et Enterprise Edition)

La liste ci-dessous résume la comptabilité des fonctions `lang` avec les langages de transformation.

lang | datetime functions

Les fonctions `lang | datetime` sont compatibles avec C#, C++, Java, Built-In.

fonctions lang | file

Les fonctions **read-binary-file** et **write-binary-file** sont uniquement compatibles avec Built-In.

fonctions lang | generator

La fonction **create-guid** est disponible pour le C#, C++, Java, Built-In.

fonctions lang | logical

Les fonctions `lang | Logique` sont disponibles pour C#, C++, Java, Built-In.

fonctions lang | math

Les fonctions `lang | math` sont disponibles pour C#, C++, Java, Built-In.

fonctions lang | QName

Les fonctions `lang | QName` sont compatibles avec C#, C++, Java, Built-In.

lang | fonctions string

- **charset-decode**, **charset-encode**: Built-In ;
- **match-pattern** : C#, Java, Built-In.
- **capitalize**, **count-substring**, **empty**, **find-substring**, **format-guid-string**, **left**, **left-trim**, **lowercase**, **pad-string-left**, **pad-string-right**, **repeat-string**, **replace**, **reversefind-substring**, **right**, **right-trim**, **string-compare**, **string-compare-ignore-case**, **uppercase**: C#, C++, Java, Built-In.

fonctions mime (uniquement MapForce Enterprise Edition)

Les fonctions `mime` sont uniquement disponibles pour Built-In.

fonctions xbrl (uniquement MapForce Enterprise Edition)

Les fonctions `xbrl` sont compatibles avec C#, C++, Java, Built-In.

fonctions xlsx (uniquement MapForce Enterprise Edition)

Les fonctions `xlsx` sont compatibles avec XSLT 2.0, XSLT 3.0, C#, Java, et Built-In.

fonctions xpath2

Toutes les fonctions `xpath2` sont compatibles avec XSLT 2.0, XSLT 3.0 et XQuery 1.0.

fonctions xpath3

Toutes les fonctions `xpath3` sont compatibles uniquement avec XSLT 3.0.

fonctions xslt10

La liste ci-dessous résume la comptabilité des fonctions `xslt10` avec les langages de transformation.
fonctions xslt10 | xpath

- **local-name**, **name**, **namespace-uri**: XSLT 1.0, XSLT 2.0, et XSLT 3.0.
- **lang**, **last**, **position**: XSLT 1.0.

fonctions xslt10 | xslt

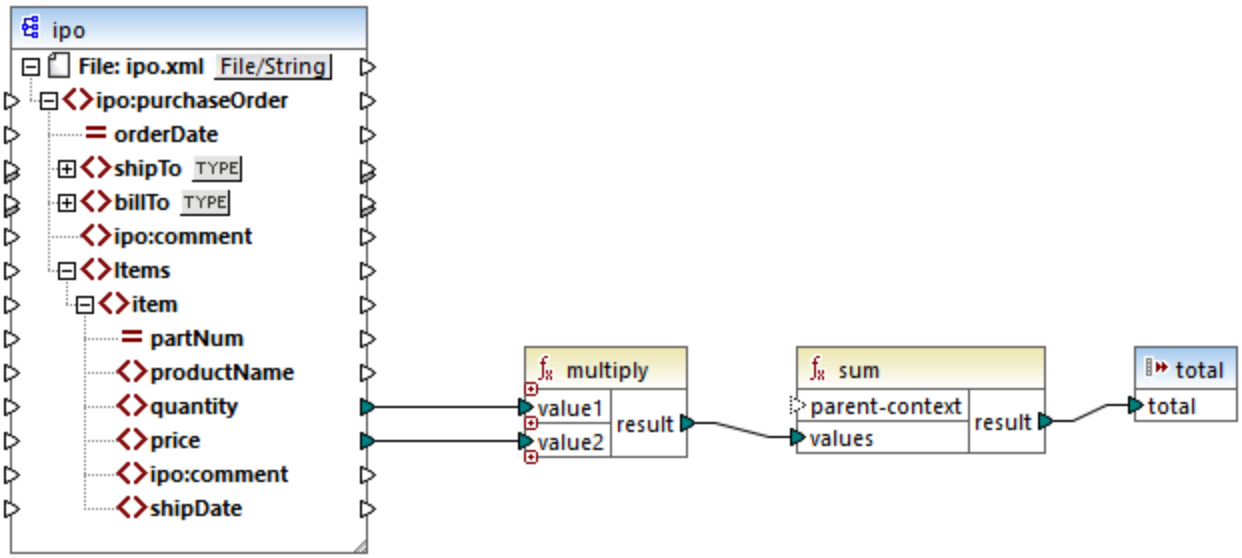
- **generate-id**, **system-property**: XSLT 1.0, XSLT 2.0, et XSLT 3.0.
- **current**, **document**, **element-available**, **function-available**, **unparsed-entity-uri**: XSLT 1.0.

6.6.1 core | aggregate functions

"Aggregate" signifie traiter plusieurs valeurs du même type de manière à obtenir un seul résultat, comme une somme, un décompte ou une moyenne. Vous pouvez effectuer des regroupements (aggregation) de données dans MapForce avec l'aide des fonctions de rassemblement, comme `avg`, `count`, `max`, etc.

Les deux arguments suivants sont communs à toutes les fonctions de rassemblement:

1. **parent-context**. Cet argument est optionnel ; il vous permet de contourner le contexte de mappage par défaut (et donc de changer l'étendue de la fonction, ou les valeurs que la fonction doit itérer). Pour voir un exemple, consulter Exemple : Changer le contexte de Parent 412.
2. **values**. Cet argument doit être connecté à un item de source qui fournit les valeurs à traiter. Par exemple, dans le mappage illustré ci-dessous, la fonction **sum** prend en entrée une séquence de valeurs numériques qui provient d'un fichier XML de source. Pour chaque item dans le fichier XML de source, la fonction **multiply** reçoit le prix de l'item multiplié par la quantité et transmet le résultat à la fonction **sum**. La fonction **sum** rassemblera toutes les valeurs d'entrée et produira un résultat total qui est également la sortie du mappage. Vous trouverez ce mappage dans le dossier **MapForceExamples**.

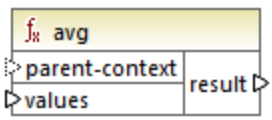


SimpleTotal.mfd

Certaines fonctions aggregate, comme `min`, `max`, `sum` et `avg`, fonctionnement exclusivement avec des valeurs numériques. Les données d'entrée de ces fonctions sont converties dans le type de données **decimal** pour traitement.

6.6.1.1 avg

Retourne la somme moyenne de toutes les valeurs dans la séquence d'entrée. La moyenne d'un set vide est un set vide.



Langages

Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

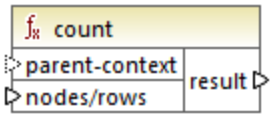
Argument	Description
parent-context	Argument optionnel. Fournit le contexte parent. Voir aussi Exemple : Changer le contexte de Parent 412.
valeurs	Cet argument doit être connecté à un item de source qui fournit les données actuelles. Veuillez noter que la valeur d'argument fournie doit être numérique.

Exemple

Voir Exemple : Regrouper les enregistrements par clé 284.

6.6.1.2 count

Retourne le nombre des items individuels constituant la séquence d'entrée. Le décompte d'un set vide est zéro.



Langages

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Veuillez noter que cette fonction a une fonctionnalité limitée dans XSLT 1.0.

Paramètres

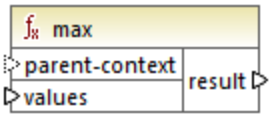
Argument	Description
parent-context	Argument optionnel. Fournit le contexte parent. Voir aussi Exemple : Changer le contexte de Parent 412. `parent-context` est un argument optionnel dans certaines fonctions d'agrégation core MapForce (comme dans **`min`**, **`max`**, **`avg`**, **`count`**). Dans un composant de source qui possède plusieurs séquences hiérarchiques, le contexte parent détermine l'ensemble de nœuds dans lequel la fonction doit fonctionner.
nodes/rows	Cet argument doit être connecté à l'item de source à compter.

Exemple

Voir Exemple : Modifier le contexte Parental 412.

6.6.1.3 max

Retourne la valeur maximum de toutes les valeurs numériques dans la séquence d'entrée. Le maximum d'un ensemble vide est un ensemble vide.



Langages

Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

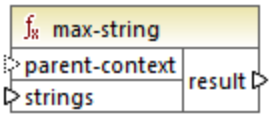
Argument	Description
parent-context	Argument optionnel. Fournit le contexte parent. Voir aussi Exemple : Changer le contexte de Parent 412. `parent-context` est un argument optionnel dans certaines fonctions d'agrégation core MapForce (comme dans **`min`**, **`max`**, **`avg`**, **`count`**). Dans un composant de source qui possède plusieurs séquences hiérarchiques, le contexte parent détermine l'ensemble de nœuds dans lequel la fonction doit fonctionner.
valeurs	Cet argument doit être connecté à un item de source qui fournit les données actuelles. Veuillez noter que la valeur d'argument fournie doit être numérique. Pour obtenir le maximum d'une séquence de strings, utiliser la fonction max-string 237.

Exemple

Voir Exemple: Regrouper les enregistrements par clé. 284

6.6.1.4 max-string

Retourne la valeur maximum de toutes les valeurs string dans la séquence d'entrée. Par exemple, **`max-string("a", "b", "c")`** retourne **`"c"`**. La fonction retourne un ensemble vide si l'argument **strings** est un ensemble vide.



Langages

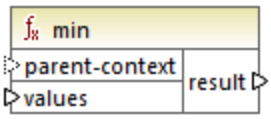
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Argument	Description
parent-context	Argument optionnel. Fournit le contexte parent. Voir aussi Exemple : Changer le contexte de Parent 412. `parent-context` est un argument optionnel dans certaines fonctions d'agrégation core MapForce (comme dans **`min`**, **`max`**, **`avg`**, **`count`**). Dans un composant de source qui possède plusieurs séquences hiérarchiques, le contexte parent détermine l'ensemble de nœuds dans lequel la fonction doit fonctionner.
strings	Cet argument doit être connecté à un item de source qui fournit les données actuelles. La valeur d'argument fourni doit être une séquence (zéro ou plusieurs) de `xs:string`.

6.6.1.5 min

Retourne la valeur minimum de toutes les valeurs numériques dans la séquence d'entrée. Le minimum d'un ensemble vide est un ensemble vide.



Langages

Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

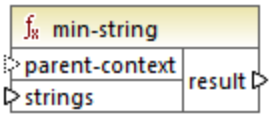
Argument	Description
parent-context	Argument optionnel. Fournit le contexte parent. Voir aussi Exemple : Changer le contexte de Parent 412. `parent-context` est un argument optionnel dans certaines fonctions d'agrégation core MapForce (comme dans **`min`**, **`max`**, **`avg`**, **`count`**). Dans un composant de source qui possède plusieurs séquences hiérarchiques, le contexte parent détermine l'ensemble de nœuds dans lequel la fonction doit fonctionner.
valeurs	Cet argument doit être connecté à un item de source qui fournit les données actuelles. Veuillez noter que la valeur d'argument fournie doit être numérique. Pour obtenir le minimum d'une séquence de strings, utiliser la fonction min-string 239.

Exemple

Voir Exemple : Regrouper les enregistrements par clé 284.

6.6.1.6 min-string

Retourne la valeur minimum de toutes les valeurs de string dans la séquence d'entrée. Par exemple, `min-string("a", "b", "c")` retourne `"a"`. La fonction retourne un ensemble vide si l'argument **strings** est un ensemble vide.



Langages

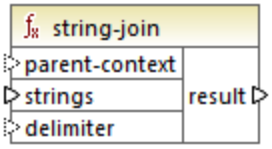
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Argument	Description
parent-context	Argument optionnel. Fournit le contexte parent. Voir aussi Exemple : Changer le contexte de Parent 412. `parent-context` est un argument optionnel dans certaines fonctions d'agrégation core MapForce (comme dans **`min`**, **`max`**, **`avg`**, **`count`**). Dans un composant de source qui possède plusieurs séquences hiérarchiques, le contexte parent détermine l'ensemble de nœuds dans lequel la fonction doit fonctionner.
strings	Cet argument doit être connecté à un item de source qui fournit les données actuelles. La valeur d'argument fourni doit être une séquence (zéro ou plusieurs) de `xs:string`.

6.6.1.7 string-join

Concatène toutes les valeurs de la séquence d'entrée dans un string délimité par le string que vous avez choisi d'utiliser en tant que le délimiteur. La fonction retourne un string vide si l'argument **strings** est un ensemble vide.



Langages

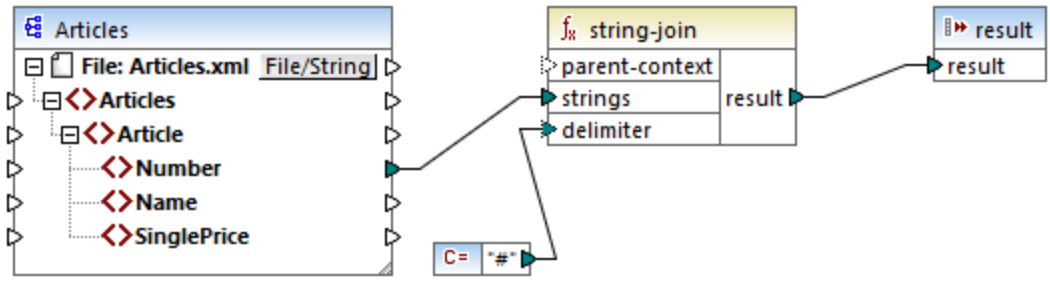
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Argument	Description
parent-context	Argument optionnel. Fournit le contexte parent. Voir aussi Exemple : Changer le contexte de Parent 412. `parent-context` est un argument optionnel dans certaines fonctions d'agrégation core MapForce (comme dans **`min`**, **`max`**, **`avg`**, **`count`**). Dans un composant de source qui possède plusieurs séquences hiérarchiques, le contexte parent détermine l'ensemble de nœuds dans lequel la fonction doit fonctionner.
strings	Cet argument doit être connecté à un item de source qui fournit les données actuelles. La valeur d'argument fourni doit être une séquence (zéro ou plusieurs) de `xs:string`.
delimiter	Argument optionnel. Spécifie le délimiteur à insérer entre deux strings consécutifs.

Exemple

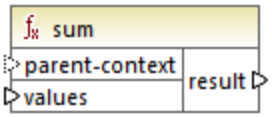
Dans l'exemple ci-dessous, le fichier XML de source contient quatre items **Article**, avec les nombres suivants : 1, 2, 3 et 4.



La constante fournit le caractère "#" en tant que délimiteur. Le résultat de mappage est donc **1#2#3#4**. Si vous ne fournissez pas de délimiteur, le résultat devient **1234**.

6.6.1.8 sum

Retourne la somme arithmétique de toutes les valeurs dans la séquence d'entrée. La somme d'un set vide est zéro.



Langues

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Paramètres

Argument	Description
parent-context	Argument optionnel. Fournit le contexte parent. Voir aussi Exemple : Changer le contexte de Parent 412. `parent-context` est un argument optionnel dans certaines fonctions d'agrégation core MapForce (comme dans **`min`**, **`max`**, **`avg`**, **`count`**). Dans un composant de source qui possède plusieurs séquences hiérarchiques, le contexte parent détermine l'ensemble de nœuds dans lequel la fonction doit fonctionner.
valeurs	Cet argument doit être connecté à un item de source qui fournit les données actuelles. Veuillez noter que la valeur d'argument fournie doit être numérique.

Exemple

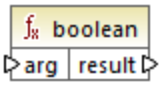
Voir Exemple : Additionner les valeurs de nœud 224.

6.6.2 core | conversion functions

Pour prendre en charge explicitement la conversion de type de données, plusieurs fonction de conversion de type sont disponibles dans la bibliothèque **conversion**. Veuillez noter que les fonctions de conversion ne sont pas toujours nécessaires parce que, dans la plupart des cas, MapForce crée les conversion nécessaires automatiquement. Les fonctions de conversion sont généralement utiles pour formater des valeurs de date et d'heures, ou pour comparer des valeurs. Par exemple, si certains items de mappage sont de types différents (comme des entiers et des strings), vous pouvez utiliser la fonction de conversion number 250 pour forcer une comparaison numérique.

6.6.2.1 boolean

Convertit la valeur de **arg** dans une valeur Booléenne Cela peut être utile pour travailler avec des fonctions logiques (comme `equal`, `greater`, etc.), ainsi qu'avec des filtres et des conditions if-else[176]. Pour obtenir une Booléenne **false**, fournir un string vide ou un numérique 0 en tant qu'argument. Pour obtenir une Booléenne **true**, fournir un string non-vide ou un numérique 1 en tant qu'argument.



Langages

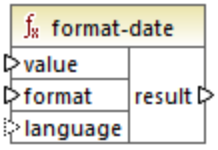
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Paramètre

Argument	Description
arg	Argument obligatoire. Fournit la valeur à convertir.

6.6.2.2 format-date

Convertit une valeur d'entrée `xs:date` dans un string et le formate conformément à des options spécifiées.



Langages

Built-in, C++, C#, Java, XSLT 2.0, XSLT 3.0.

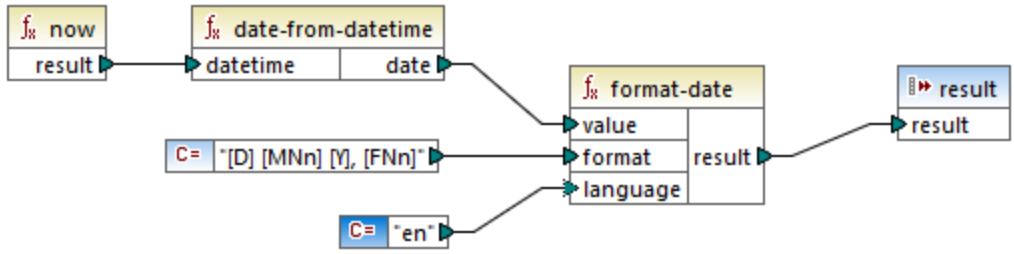
Paramètres

Argument	Description
value	La valeur `xs:date` à formater.
format	Un string de format identifiant la manière avec laquelle la date doit être formatée. Cet argument est utilisé de la même manière que l'argument **format** dans la fonction format-dateTime[243].

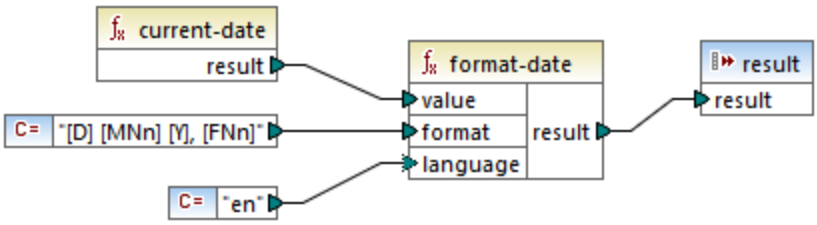
Argument	Description
language	Argument optionnel. Lorsqu'il est fourni, le nom du mois est le jour de la semaine sont retournés dans un langage spécifique. Valeurs valides : **de** Allemand **en (défaut)** Anglais **es** Espagnol **fr** Français **ja** Japonais

Exemple

Le mappage suivant sort la date actuel dans un format comme : "25 March 2020, Wednesday". Pour traduire cette valeur en espagnol, définir la valeur de l'argument de **language** sur **es**.

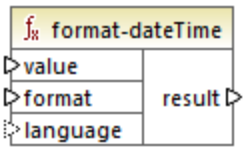


Veuillez noter que le mappage ci-dessus est conçu pour les langages de transformation Built-in, C++, C# ou Java. Dans XSLT 2.0, le même résultat peut être obtenu par le mappage suivant :



6.6.2.3 format-dateTime

Convertit une valeur de type `xs:dateTime` en un string. La représentation string de date et time est formatté conformément à la valeur de l'argument **format**.



Langages

Built-in, C++, C#, Java, XSLT 2.0, XSLT 3.0.

Paramètres

Argument	Description
value	La valeur `xs:dateTime` à formater.
format	Un string de format identifiant la manière avec laquelle **value** doit être formatée. Voir "Remarques" ci-dessous.
language	Argument optionnel. Lorsqu'il est fourni, le nom du mois est le jour de la semaine sont retournés dans un langage spécifique. Valeurs valides : **de** Allemand **en (défaut)** Anglais **es** Espagnol **fr** Français **ja** Japonais

Note : Si la sortie de la fonction (résultat) est connectée à un item de type différent d'un string, le formatage peut être perdu lorsque la valeur est amenée vers le type de cible. Pour désactiver ce cast automatique, supprimer les cases à cocher **Valeurs cibles cast vers les types de cible** dans les Paramètres de composant 39 du composant cible.

Remarques

L'argument **format** consiste en un string contenant de soit-disant marqueurs de variable contenue dans des crochets, par exemple `[Y]/[M]/[D]`. Les caractères se trouvant en dehors des crochets sont des caractères littéraux. Si les crochets sont nécessaires en tant que caractères littéraux dans le résultat, alors ils devraient être doublés.

Chaque marqueur de variable consiste en un spécificateur de composant identifiant lequel des composants de date ou time doit être affiché, un modificateur de formatage optionnel, un autre modificateur de présentation optionnel et un modificateur de largeur optionnel, précédé par une virgule si elle est présente.

```
format := (literal | argument)*
argument := [component(format)?(presentation)?(width)?]
width := , min-width ("-" max-width)?
```

Les composants sont les suivants :

Spécificateur	echo Hello, World!	Présentation par défaut
Y	année (valeur absolue)	quatre chiffres (2010)
M	mois de l'année	1-12
J	jour du mois	1-31
d	jour de l'année	1-366
F	jour de la semaine	nom du jour (dépendant du langage)
W	semaine de l'année	1-53
w	semaine du mois	1-5
H	heure (24 heures)	0-23
h	heure (12 heures)	1-12
P	A.M. ou P.M.	alphabétique (selon le langage)
m	minutes dans l'heure	00-59
s	secondes dans la minute	00-59
f	secondes fractionnelles	numérique, une place décimale
Z	fuseau horaire en tant qu'offset d'heure depuis UTC	+08:00
z	fuseau horaire en tant qu'offset d'heure en utilisant GMT	GMT+n

Le modificateur de format peut être un des suivants :

Caractère	echo Hello, World!	Exemple
1	Format numérique décimal sans zéros au début	1, 2, 3
01	Format numérique, deux chiffres	01, 02, 03
N	Nom du composant , casse majuscule[1]	MONDAY, TUESDAY
n	Nom du composant, casse minuscule[1]	monday, tuesday
Nn	Nom du composant, casse de titre[1]	Monday, Tuesday

Notes de bas de page :

1. Les modificateurs **N**, **n** et **Nn** sont pris en charge par les composants suivants uniquement : **M**, **d**, **D**.

Le modificateur de largeur, si nécessaire, est introduit par une virgule, suivi par un chiffre qui exprime la largeur minimum. En option, vous pouvez ajouter une barre oblique suivie par un autre chiffre qui exprime la largeur maximum. Par exemple :

- `[D,2]` est le jour du mois, avec des zéros en début de ligne (deux chiffres).
- `[MNn,3-3]` est le nom du mois, écrit avec trois lettres, par ex. *Jan*, *Feb*, *Mar*, etc.

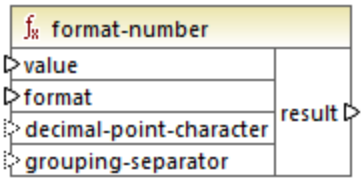
Exemples

La table ci-dessous illustre quelques exemples de valeurs de formatage `xs:dateTime` avec l'aide de la fonction **`format-dateTime`**. La colonne "Value" spécifie la valeur fournie à l'argument **value**. La colonne "Format" spécifie la valeur de l'argument **format**. La colonne "Result" illustre ce qui est retourné par la fonction.

Valeur	Format	Résultat
2003-11-03T00:00:00	`[D]/[M]/[Y]`	11/03/2003
2003-11-03T00:00:00	`[Y]-[M,2]-[D,2]`	03/11/2003
2003-11-03T00:00:00	`[Y]-[M,2]-[D,2] [H,2]:[m]:[s]`	2003-11-03 00:00:00
2010-06-02T08:02	`[Y] [MNn] [D01] [F,3-3] [d] [H]:[m]:[s].[f]`	2010 June 02 Wed 153 8:02:12.054
2010-06-02T08:02	`[Y] [MNn] [D01] [F,3-3] [d] [H]:[m]:[s].[f] [z]`	2010 June 02 Wed 153 8:02:12.054 GMT+02:00
2010-06-02T08:02	`[Y] [MNn] [D1] [F] [H]:[m]:[s].[f] [Z]`	2010 June 2 Wednesday 8:02:12.054 +02:00
2010-06-02T08:02	`[Y] [MNn] [D] [F,3-3] [H01]:[m]:[s]`	2010 June 2 Wed 08:02:12

6.6.2.4 format-number

Convertit un nombre en un string et le formate conformément aux options spécifiées.



Langages

Built-in, C++, C#, Java, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Paramètres

Argument	Description
value	Argument obligatoire. Fournit le nombre à formater.
format	Argument obligatoire. Fournit un string de format qui identifie la manière avec laquelle le nombre doit être formaté. Voir "Remarques" ci-dessous.
decimal-point-format	Argument optionnel. Fournit le caractère à utiliser en tant que le caractère de point décimal. La valeur par défaut est le caractère de point final (.).
grouping-separator	Argument optionnel. Fournit le caractère utilisé pour séparer des groupes de nombres. La valeur par défaut est le caractère de virgule (,).

Note : Si la sortie de la fonction (résultat) est connectée à un item de type différent d'un string, le formatage peut être perdu lorsque la valeur est amenée vers le type de cible. Pour désactiver ce cast automatique, supprimer les cases à cocher **Valeurs cibles cast vers les types de cible** dans les Paramètres de composant[39] du composant cible.

Remarques

L'argument **format** prend la forme suivante :

```
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)
```

Le premier *subformat* est utilisé pour formater des nombres positifs, et le second subformat pour des nombres négatifs. Si un seul *subformat* est spécifié, alors le même sous-format sera utilisé pour des nombres négatifs, mais avec un signe négatif rajouté avant le *prefix*.

Caractère spécial	Défaut	echo Hello, World!
zero-digit	0	Un nombre apparaîtra toujours à cet endroit du résultat
digit	#	Un nombre apparaîtra à cet endroit du string de résultat à moins qu'il s'agisse d'un zéro de début ou de fin redondant
decimal-point	.	Sépare l'entier et la partie fractionnelle du nombre
grouping-separator	,	Sépare des groupes de chiffres.

Caractère spécial	Défaut	echo Hello, World!
percent-sign	%	Multiplie le nombre par 100 et le montre en tant que pourcentage.
per-mille	‰	Multiplie le nombre par 1000 et le montre en tant que pourmille.

La table ci-dessous illustrate des exemples des strings de format et leur résultat.

Note: La méthode d'arrondissement utilisée par la fonction `format-number` est "half up", ce qui signifie que la valeur est arrondie si la fraction est supérieure ou égale à 0.5 La valeur est arrondie vers le bas si la fraction est inférieure à 0.5. Cette méthode d'arrondissement s'applique uniquement au code de programme généré et le moteur d'exécution Built-In. Dans XSLT 1.0, le mode d'arrondissement est non-défini. Dans XSLT 2.0, le mode d'arrondissement est "round-half-to-even".

Numéro	String de Format	Résultat
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

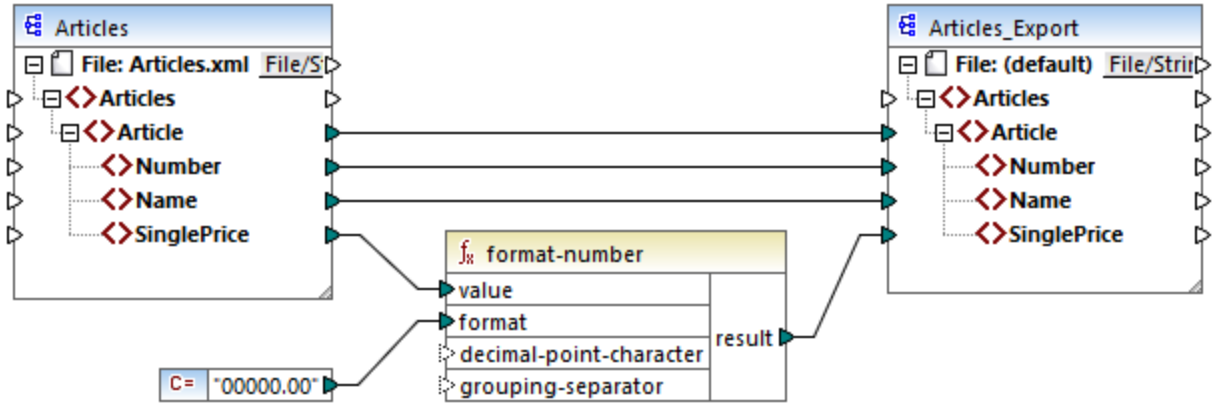
Exemple

Le mappage illustré ci-dessous lit des données provenant d'un XML de source et l'écrit dans un XML cible. Il existe plusieurs éléments **SinglePrice** dans la source qui contient les valeurs décimales suivantes : **25**, **2.30**, **34**, **57.50**. Le mappage a deux objectifs :

1. Remplir toutes les valeurs avec des zéros à gauche de manière à ce que la partie importante prenne exactement 5 chiffres

2. Remplir toutes les valeurs avec des zéros à droite de manière à ce que la partie importante prenne exactement 2 chiffres

Pour ce faire, le string de format `00000.00` a été fourni en tant qu'argument à la fonction `format-number`.



PreserveFormatting.mfd

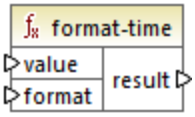
Par conséquent, les valeurs dans la cible sont devenues :

```
00025,00
00002,30
00034,00
00057,50
```

Vous pouvez trouver le fichier de design de mappage sous le chemin suivant:
<Documents>\Altova\MapForce2024\MapForceExamples\PreserveFormatting.mfd.

6.6.2.5 format-time

Convertit la valeur d'entrée `xs:time` dans un string.



Langages

Built-in, C++, C#, Java, XSLT 2.0, XSLT 3.0.

Paramètres

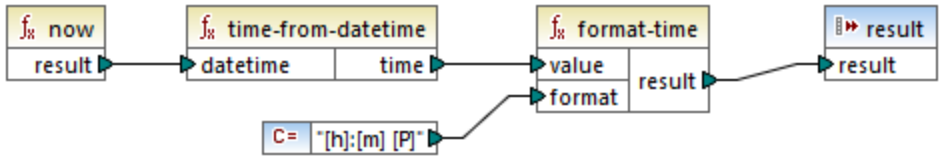
Argument	Description
value	Argument obligatoire. Fournit la valeur `xs:time` à formater.

Argument	Description
format	Argument obligatoire. Fournit un string de format. Cet argument est utilisé de la même manière que l'argument **format** dans la fonction format-dateTime 243.

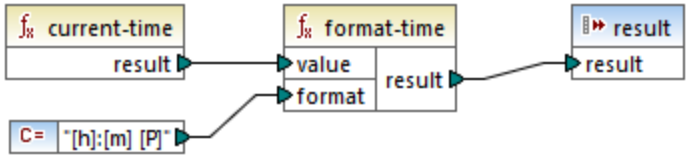
Exemple

Le mappage suivant produit l'heure actuelle dans un format comme `2:15 p.m.` . Pour y parvenir, il utilise le string de format `[h]:[m] [P]`, où :

- **[h]** est l'heure actuelle dans un format 12 heures
- **[m]** est la minute actuelle
- **[P]** est la partie "a.m." ou "p.m."

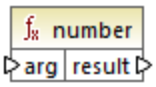


Veuillez noter que le mappage ci-dessus est conçu pour les langages de transformation Built-in, C++, C# ou Java. Dans XSLT 2.0, le même résultat peut être obtenu par le mappage suivant :



6.6.2.6 number

Convertit la valeur de **arg** en un nombre, où **arg** est un string ou une valeur Booléenne. Si **arg** est un string, MapForce tentera de le parser en tant que nombre. Par exemple, un string comme `"12.56 »` est converti dans la valeur décimale `12.56`. Si **arg** est booléenne **true**, est converti dans le numérique **1**. Si **arg** est booléenne **false**, est converti dans le numérique **0**.



Langages

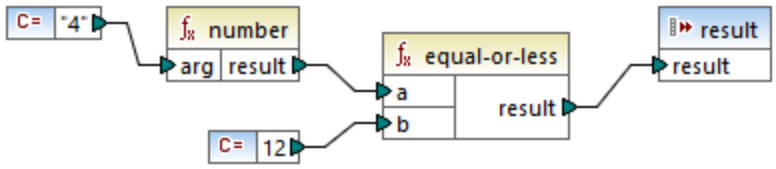
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Paramètres

Argument	Description
arg	Argument obligatoire. Fournit la valeur à convertir.

Exemple

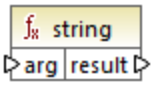
Dans l'exemple ci-dessous, la première constante est de type `string` et elle contient le string "4". La seconde constante contient la constante numérique 12. Pour que les deux valeurs puissent être comparées en tant que nombres, les types doivent s'accorder.



En ajoutant une fonction **number** à la première constante convertir le string "4" dans la valeur numérique de 4. Le résultat de la comparaison est alors "true". Si la fonction **number** n'a pas été utilisée (c'est à dire, si "4" a été connecté directement à **a**), une comparaison de string se produirait, le résultat étant "false".

6.6.2.7 string

Convertit la valeur d'entrée dans un string. La fonction peut aussi être utilisée pour extraire le contenu de texte d'un nœud. Si le nœud d'entrée est un types complexe XML, alors tous les descendants sont aussi sortis en tant que string unique.



Langages

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Paramètres

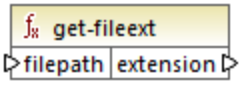
Argument	Description
arg	Argument obligatoire. Fournit la valeur à convertir.

6.6.3 core | file path functions

Les fonctions **file path** vous permettent d'accéder directement et de manipuler des données de chemin de fichier, comme des dossiers, des noms de fichier et des extensions pour un traitement ultérieur dans vos mappages. Elles peuvent être utilisées dans tous les langages pris en charge par MapForce.

6.6.3.1 get-fileext

Retourne l'extension du chemin de fichier contenant le caractère ".".



Langages

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Paramètres

Argument	Description
filepath	Argument obligatoire Fournit le chemin de fichier à traiter.

Exemple

Si vous fournissez "c:\data\Sample.mfd" en tant qu'argument, le résultat est `.mfd`.

6.6.3.2 get-folder

Retourne le nom du dossier du chemin de fichier y compris la barre oblique de fin, ou la barre oblique inversée.

Langages

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Paramètres

Argument	Description
filepath	Argument obligatoire Fournit le chemin de fichier à traiter.

Exemple

Si vous fournissez "c:\data\Sample.mfd" en tant qu'argument, le résultat est **`c:\data\`**.

6.6.3.3 main-mfd-filepath

Retourne le chemin complet du fichier de design de mappage (.mfd) contenant le mappage principal. Un string vide est retourné si le .mfd n'est pas enregistré actuellement.

Langages

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

6.6.3.4 mfd-filepath

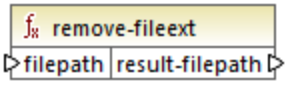
Si la fonction est appelée dans le mappage principal, elle retourne la même chose que la fonction main-mfd-filepath 253 , c.à.d. le chemin complet du fichier .mfd contenant le mappage principal. Un string vide est retourné si le fichier . mfd n'est actuellement pas enregistré. Si appelé dans le cadre d'une fonction définie par l'utilisateur qui est *importée* par un mfdfile, il retourne le chemin complet du fichier mfd *importé* qui contient la définition de la fonction définie par l'utilisateur.

Langages

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

6.6.3.5 remove-fileext

Supprime l'extension du chemin de fichier contenant le caractère dot.



Langages

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Paramètres

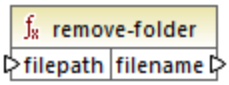
Argument	Description
filepath	Argument obligatoire Fournit le chemin de fichier à traiter.

Exemple

Si vous fournissez "c:\data\Sample.mfd" en tant qu'argument, le résultat est **`c:\data\Sample`**.

6.6.3.6 remove-folder

Supprime le répertoire du chemin de fichier contenant le caractère de la barre oblique de fin, ou la barre oblique inversée.



Langages

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Paramètres

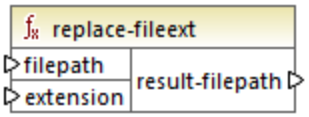
Argument	Description
filepath	Argument obligatoire Fournit le chemin de fichier à traiter.

Exemple

Si vous fournissez "c:\data\Sample.mfd" en tant qu'argument, le résultat est **`Sample.mfd`**.

6.6.3.7 replace-fileext

Remplace l'extension du chemin de fichier fourni par le paramètre **filepath** celui fourni par la connexion du paramètre d**extension**.



Langages

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Paramètres

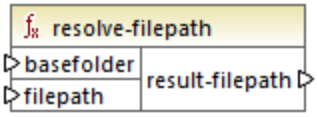
Argument	Description
filepath	Argument obligatoire Fournit le chemin de fichier à traiter.
extension	Argument obligatoire Fournit la nouvelle extension à utiliser.

Exemple

Si vous fournissez "c:\data\Sample.log" en tant que **filepath**,et ".txt" en tant qu'**extension**, le résultat est `c:\data\Sample.txt`.

6.6.3.8 resolve-filepath

Résoud un chemin de fichier relatif contre un dossier de base. La fonction prend en charge '.' (répertoire actuel) et '..' (répertoire parent).



Langages

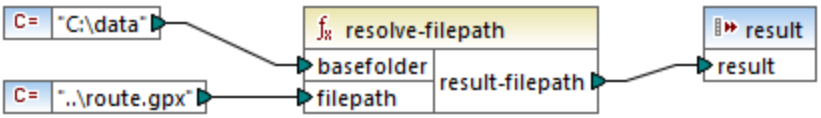
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Paramètres

Argument	Description
basefolder	Argument obligatoire Fournit le répertoire de base relatif au chemin qui doit être résolu. Il peut s'agir d'un chemin absolu ou relatif.
filepath	Argument obligatoire Fournit le fichier relatif à résoudre.

Exemples

Dans le mappage ci-dessous, le chemin de fichier relatif **..\route.gpx** est résolu par rapport au répertoire **C:\data**.



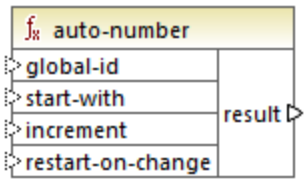
Le résultat de mappage est **`C:\route.gpx`**.

6.6.4 core | generator functions

La bibliothèque de fonctions **core / generator** contient des fonctions qui génèrent des valeurs.

6.6.4.1 auto-number

Génère des nombres entiers dans une séquence (par exemple, 1,2,3,4, ...). Il est possible de définir l'entier de démarrage, la valeur d'incrémentation et d'autres options par le biais des paramètres.



L'ordre exact dans lequel des fonctions sont appelées par le code de mappage généré n'est pas défini. MapForce peut nécessiter de dissimuler des résultats calculés à réutiliser ou évaluer des expressions dans n'importe quel ordre. De même, contrairement à d'autres fonctions, la fonction **`auto-number`** retourne un résultat différent lorsqu'ils sont appelés plusieurs fois avec les même paramètres d'entrée. C'est pourquoi il est fortement recommandé d'utiliser la fonction **`auto-number`** avec précaution. Dans certains cas, il est possible d'obtenir le même résultat en utilisant la fonction position 297 à la place.

Langages

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Paramètres

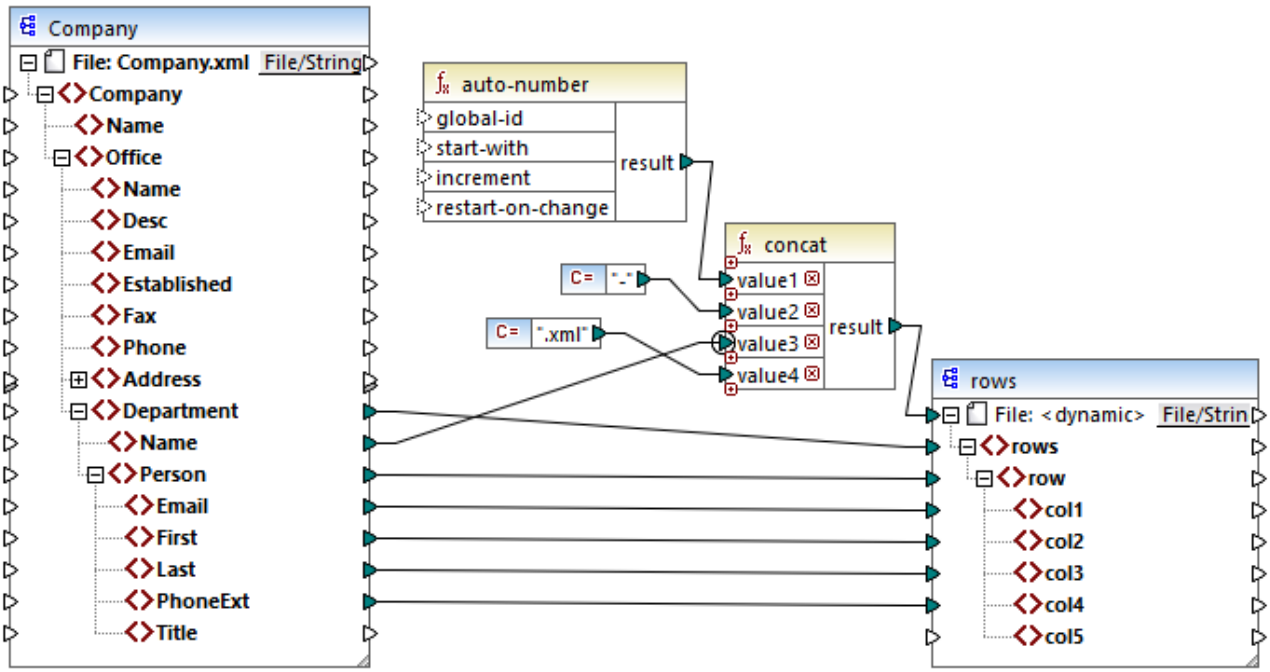
Argument	Description
global-id	Paramètre optionnel. Si un design de mappage contient plusieurs fonctions **`auto-number`**, elles généreront des séquences avec des nombres dupliqués (se chevauchant). Pour rendre toutes les fonctions **`auto-number`** conscientes l'une de l'autre, et donc générer des séquences qui ne se chevauchent pas, connecter un string commun (par exemple une constante) dans l'entrée **global-id** de chaque fonction **`auto-number`**.

Argument	Description
start-with	Paramètre optionnel. Spécifie l'entier avec lequel la séquence générée commence. La valeur par défaut est de **1**.
increment	Paramètre optionnel. Spécifie la valeur d'incrément. La valeur par défaut est de **1**.
restart-on-change	Paramètre optionnel. Réinitialise le décompte sur **start-with**, lorsque le contenu de l'item connecté change.

Exemple

Le mappage suivant est une variation du mappage **ParentContext.mfd** discuté dans l'Exemple : Changer le contexte de Parent 412.

L'objectif du mappage illustré ci-dessous est de générer plusieurs fichiers XML, un pour chaque département dans le fichier XML de source. Il existe des département portant le même nom (parce qu'ils appartiennent à des bureaux de parents différents). Pour cette raison, chaque nom de fichier généré doit commencer avec un nombre séquentiel, par exemple **1-Administration.xml**, **2-Marketing.xml**, etc.



Pour atteindre l'objectif de mappage, la fonction **`auto-number`** a été utilisée. Le résultat de cette fonction est concaténé avec un caractère de tiret, suivi par le nom du département, suivi par le string ".xml" afin de créer le nom unique du fichier généré. Chose importante, le troisième paramètre de la fonction **`concat`** (le nom de département) a un priority context 416 qui s'applique. Cela a pour conséquence que la fonction **`auto-number`** est appelée dans le contexte de chaque département et produit les valeurs séquentielles requises. Si le contexte de priorité n'a pas été utilisé, la fonction **`auto-number`** garderait le nombre 1 (dans l'absence de tout contexte), et des noms de fichier doubles seraient générées en conséquence.

6.6.5 core | logical functions

Les fonctions logiques sont (généralement) utilisées pour comparer des données d'entrée avec le résultat étant une valeur booléenne `true` ou `false`. Elles sont généralement utilisées pour tester des données avant d'être transmises à un sous-ensemble vers le composant de cible à l'aide d'un filtre[176]. Presque toutes les fonctions logiques ont la structure suivante :

- paramètres d'entrée : `a | b` ou `value1 | value2`
- paramètres de sortie : `résultat`

Le résultat d'évaluation dépend des valeurs d'entrée et des types de données utilisés pour la comparaison. Par exemple, la comparaison 'inférieur à' des valeurs d'entier **4** et **12** donne la valeur booléenne `true`, étant donné que 4 est inférieur à 12. Si les deux paramètres d'entrée contiennent les valeurs de string **4** et **12**, l'analyse lexicale résulte dans la valeur de sortie `false`, puisque **4** est alphabétiquement supérieur au premier caractère **1** du second opérande (**12**).

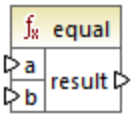
Si toutes les valeurs d'entrée sont de même type de données, alors la comparaison est effectuée pour le type commun. Si des valeurs d'entrée sont de types différents (par exemple, `integer` et `string` ou `string` et `date`), alors le type de données utilisé pour la comparaison est le plus général (le moins restrictif) des deux.

Avant de comparer deux valeurs de types différents, toutes les valeurs d'entrée sont converties en un type de données commun. En reprenant l'exemple précédent ; le type de données `string` est moins restrictif que `integer`. Comparer la valeur d'entier **4** avec le string **12** convertit la valeur d'entier **4** vers le string **4**, qui est ensuite comparé avec le string **12**.

Note: Les fonctions logiques ne peuvent pas être utilisées pour tester l'existence de valeurs nulles. Si vous fournissez une valeur nulle en tant qu'argument pour une fonction logique, elle retourne une valeur nulle. Pour plus d'informations concernant les valeurs nulles, voir Nil Values / Nillable[120].

6.6.5.1 equal

La fonction **`equal`** (*voir la capture d'écran ci-dessous*) retourne Booléenne `true` si **a** est la même que **b** ; autrement `false`. La comparaison est sensible à la casse.



Exemple :

```
a = hi
b = hi
```

Dans cet exemple, les deux valeurs sont les mêmes. Pour cette raison, le résultat est `true`. Si, par exemple, `b` égalisait `Hi`, la fonction retournerait `false`.

Langages

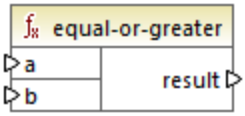
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Paramètres

Argument	Description
a	Paramètre obligatoire. Fournit la première valeur à comparer.
b	Paramètre obligatoire. Fournit la seconde valeur à comparer.

6.6.5.2 equal-or-greater

Retourne Booléenne **true** si *a* est égal ou supérieur à *b*; **false** sinon.



Langages

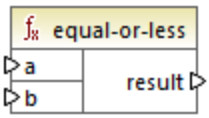
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Paramètres

Argument	Description
a	Paramètre obligatoire. Fournit la première valeur à comparer.
b	Paramètre obligatoire. Fournit la seconde valeur à comparer.

6.6.5.3 equal-or-less

Retourne Booléenne **true** si *a* est égal ou inférieur à *b*; **false** sinon.



Langages

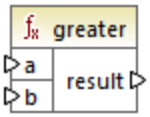
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Paramètres

Argument	Description
a	Paramètre obligatoire. Fournit la première valeur à comparer.
b	Paramètre obligatoire. Fournit la seconde valeur à comparer.

6.6.5.4 greater

Retourne Booléenne **true** si *a* est supérieur à *b*; **false** sinon.



Langages

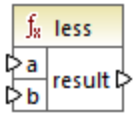
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Paramètres

Argument	Description
a	Paramètre obligatoire. Fournit la première valeur à comparer.
b	Paramètre obligatoire. Fournit la seconde valeur à comparer.

6.6.5.5 less

Retourne Booléenne **true** si *a* est inférieur à *b*; **false** sinon.



Langages

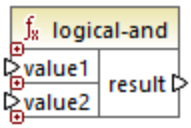
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Paramètres

Argument	Description
a	Paramètre obligatoire. Fournit la première valeur à comparer.
b	Paramètre obligatoire. Fournit la seconde valeur à comparer.

6.6.5.6 logical-and

Retourne Booléenne **true** uniquement si chaque valeur d'entrée est true ; **false** sinon. Vous pouvez connecter le résultat à une autre fonction **`logical-and`** et rejoindre un nombre arbitraire de conditions avec la logique AND, afin de tester qu'elles retournent toutes **true**. De même, cette fonction peut être élargie pour prendre des arguments supplémentaires, voir Ajouter ou supprimer des arguments de fonction 196.



Langages

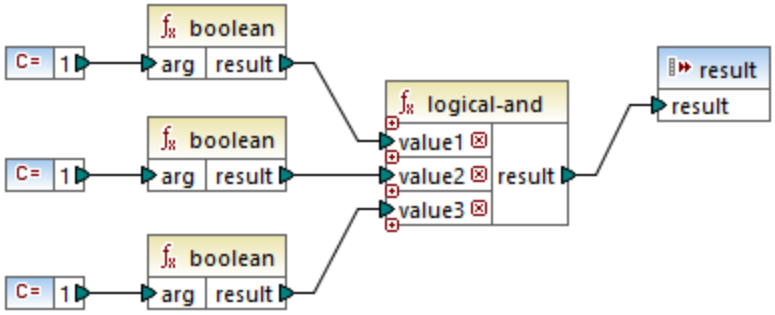
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Paramètres

Argument	Description
value1	Paramètre obligatoire. Fournit la première valeur à comparer.
value2	Paramètre obligatoire. Fournit la seconde valeur à comparer.

Exemple

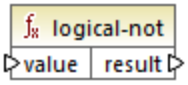
Le mappage illustré ci-dessous retourne **true** parce que toutes les valeurs d'entrée dans la fonction **`logical-and`** sont **true** également. Si une de ces valeurs d'entrée était **false**, le résultat de mappage serait **false** également.



Voir aussi Exemple : Consultation et Concaténation[216].

6.6.5.7 logical-not

Invertit ou renverse le résultat logique de la valeur d'entrée. Par exemple, si *value* est **true**, le résultat de la fonction est faux. Si *value* est **false**,alors le résultat est true



Langages

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

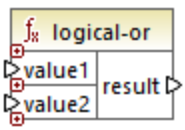
Paramètres

Argument	Description
value	Paramètre obligatoire. Fournit la valeur d'entrée.

6.6.5.8 logical-or

Cette fonction exige que les deux valeurs d'entrée soient booléennes. Si au moins une des valeurs d'entrée est **true**, alors le résultat est **true**. Sinon le résultat est **false**.

Cette fonction peut être étendue pour prendre des arguments supplémentaires, voir Ajouter ou supprimer des arguments de fonctions[196].



Langages

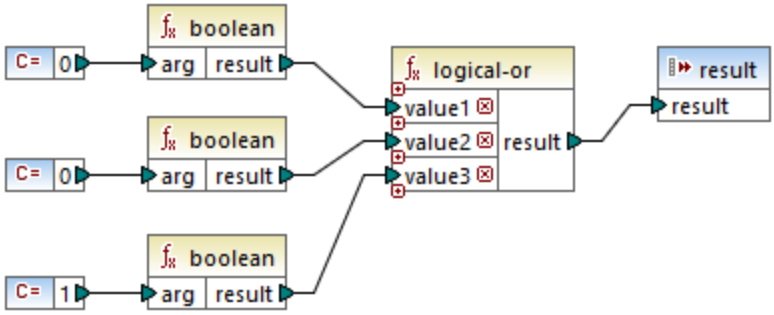
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Paramètres

Argument	Description
value1	Paramètre obligatoire. Fournit la première valeur à comparer.
value2	Paramètre obligatoire. Fournit la seconde valeur à comparer.

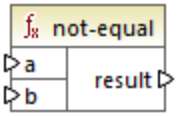
Exemple

Le résultat du mappage ci-dessous est **true**, parcequ'au moins un des arguments de la fonction est **true**.



6.6.5.9 not-equal

Retourne booléenne **true** si *a* n'est pas égal à *b*; **false** sinon.



Langages

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

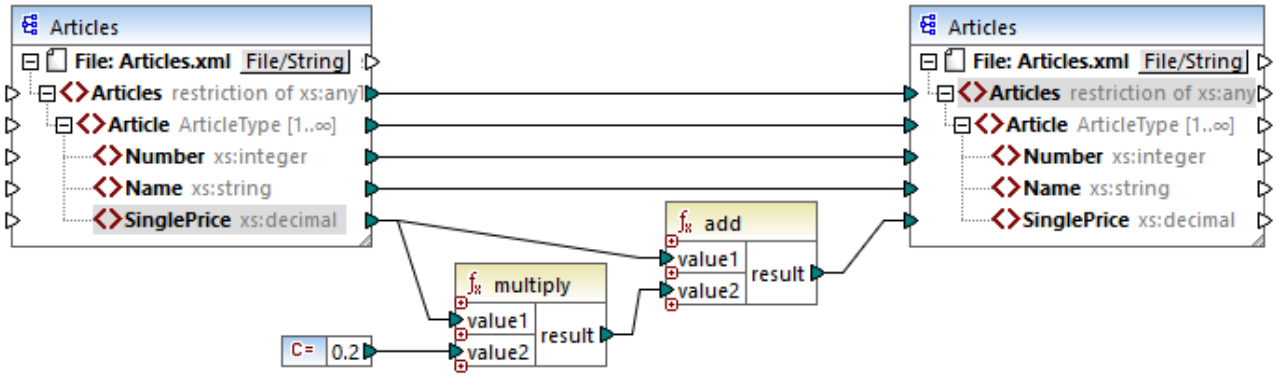
Paramètres

Argument	Description
a	Paramètre obligatoire. Fournit la première valeur à comparer.
b	Paramètre obligatoire. Fournit la seconde valeur à comparer.

6.6.6 core | math functions

Les fonctions math sont utilisées pour effectuer des opérations mathématiques de base sur des données. Veuillez noter qu'elles ne peuvent pas être utilisées pour effectuer des calculs sur les durées ou des valeurs `datetime`.

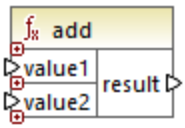
La plupart des fonctions math prennent deux paramètres d'entrée (**value1**, **value2**) qui sont des opérandes de l'opération mathématique. Les valeurs d'entrée sont converties automatiquement au type `decimal` pour un traitement ultérieur. Le résultat des fonctions math est aussi de type `decimal`.



L'exemple indiqué ci-dessus ajoute une taxe de vente de 20% pour chacun des articles mappés dans le composant de cible.

6.6.6.1 add

Ajoute **value1** à **value2** et retourne le résultat en tant que valeur décimale. Cette fonction peut être étendue pour prendre des arguments supplémentaires, voir Ajouter ou supprimer des arguments de fonctions 196.



Langages

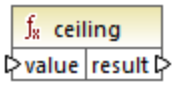
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Paramètres

Argument	Description
value1	Paramètre obligatoire. Fournit le premier opérande.
value2	Paramètre obligatoire. Fournit la seconde opérande.

6.6.6.2 ceiling

Retourne l'entier le plus petit qui est supérieur à ou égal à **value**.



Langages

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Paramètres

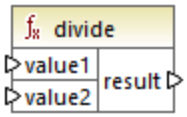
Argument	Description
value	Paramètre obligatoire. Fournit la valeur d'entrée de la fonction.

Exemple

Si la valeur d'entrée est **11.2**, y appliquer la fonction `ceiling` pour faire le résultat **12**, c.à.d. l'entier le plus petit qui est supérieur à **11.2**.

6.6.6.3 divide

Divise **value1** à **value2** et retourne le résultat en tant que valeur décimale. La précision du résultat dépend du langage cible. Utiliser la fonction round-precision 268 pour définir la précision du résultat.



Langages

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Paramètres

Argument	Description
value1	Paramètre obligatoire. Fournit le premier opérande.
value2	Paramètre obligatoire. Fournit la seconde opérande.

6.6.6.4 floor

Retourne l'entier le plus grand qui est inférieur à ou égal à **value**.

Langages

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Paramètres

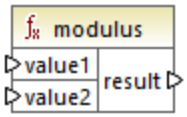
Argument	Description
value	Paramètre obligatoire. Fournit la valeur d'entrée de la fonction.

Exemple

Si la valeur d'entrée est **11.7**, y appliquer la fonction `floor` pour faire le résultat **11**, c.à.d. l'entier le plus grand qui est inférieur à **11.7**.

6.6.6.5 modulus

Retourne le reste de la division **value1** par **value2**.



Langages

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Paramètres

Argument	Description
value1	Paramètre obligatoire. Fournit le premier opérande.
value2	Paramètre obligatoire. Fournit la seconde opérande.

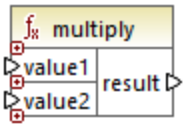
Exemple

Si les valeurs d'entrée sont **1.5** et **1**, alors le résultat de la fonction **modulus** est **0.5**. L'explication est que `1.5 / 1` laisse un restant de **0.5**.

Si les valeurs d'entrée sont **9** et **3**, alors le résultat est **0**, puisque `9 / 3` ne laisse aucun reste.

6.6.6.6 multiply

Multiplie **value1** par **value2** et retourne le résultat en tant que valeur décimale.



Langages

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Paramètres

Argument	Description
value1	Paramètre obligatoire. Fournit le premier opérande.
value2	Paramètre obligatoire. Fournit la seconde opérande.

6.6.6.7 round

Retourne la valeur arrondie à l'entier le plus proche. Lorsque la valeur se trouve exactement entre deux entiers, l'algorithme "Round Half Towards Positive Infinity" est utilisé. Par exemple, la valeur "10.5" est arrondie à "11", et la valeur "-10.5" est arrondie à "-10".

Langages

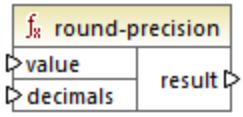
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Paramètres

Argument	Description
value	Paramètre obligatoire. Fournit la valeur d'entrée de la fonction.

6.6.6.8 round-precision

Arrondit la valeur d'entrée à N *N* décimales, où *N* est l'argument **decimals**.



Langages

Built-in, C++, C#, Java.

Paramètres

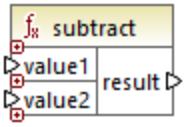
Argument	Description
value	Paramètre obligatoire. Fournit la valeur d'entrée de la fonction.
decimals	Paramètre obligatoire. Spécifie le nombre des décimales pour arrondir.

Exemple

Arrondir la valeur **2.777777** à deux décimales **2.78**. Arrondir la valeur **0.1234** à 3 décimales **0.123**.

6.6.6.9 subtract

Soustrait **value2** à **value1** et retourne le résultat en tant que valeur décimale.



Langages

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Paramètres

Argument	Description
value1	Paramètre obligatoire. Fournit le premier opérande.
value2	Paramètre obligatoire. Fournit la seconde opérande.

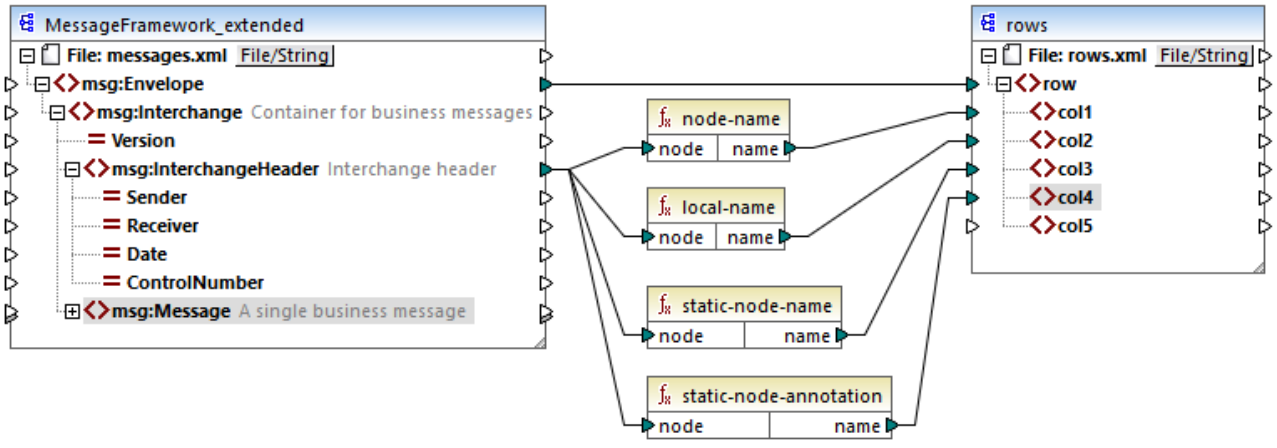
6.6.7 core | node functions

Les fonctions provenant de la bibliothèque **core | node functions** vous permettent d'accéder à des informations d'accès à propos des nœuds sur un composant de mappage (comme le nom de nœud ou l'annotation), ou pour traiter des éléments nillables, voir aussi Nil Values / Nillable 120.

Veuillez considérer qu'il existe un moyen alternatif d'accéder aux noms de nœud, qui ne nécessite pas du tout de fonctions de nœud, voir Mapper des noms de nœud 383.

Le mappage illustré ci-dessous montre quelques fonctions de nœud qui obtiennent l'information depuis le nœud **msg:InterchangeHeader** du fichier XML de source. Plus spécifiquement, les informations suivantes sont extraites :

1. La fonction **node-name** retourne le nom qualifié du nœud, qui inclut le préfixe du nœud.
2. La fonction **local-name** retourne uniquement la partie locale.
3. La fonction **static-node-name** est semblable à la fonction **node-name**, mais elle est disponible également dans XSLT 1.0.
4. La fonction **static-node-annotation** obtient l'annotation de l'élément tel qu'il a été défini dans le schéma he XML.



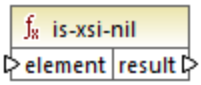
La sortie du mappage est le suivant (excluant les déclarations XML et d'espace de noms) :

```
<row>
   <col1>msg:InterchangeHeader</col1>
   <col2>InterchangeHeader</col2>
```

```
   <col3>msg:InterchangeHeader</col3>
   <col4>Interchange header</col4>
</row>
```

6.6.7.1 is-xsi-nil

Retourne **true** si le nœud **element** a l'attribut `xsi:nil` défini sur **true**.



Langages

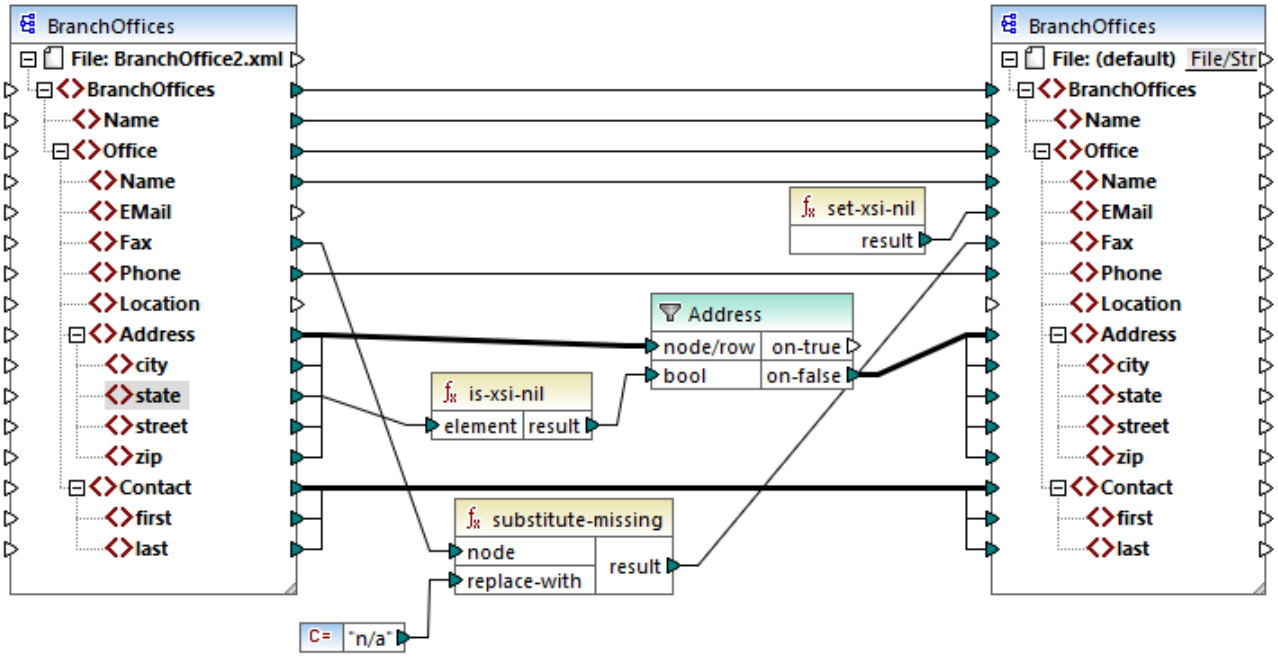
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Paramètres

Argument	Description
element	Paramètre obligatoire. Doit être connecté au nœud de source qui doit être vérifié.

Exemple

Le design de mappage illustré ci-dessous copie conditionnellement des données depuis une source vers un fichier XML de cible, et illustre également l'utilisation de plusieurs fonctions, y compris **`is-xsi-nil`**. Ce mappage est appelé **HandlingXsiNil.mfd** et peut être trouvé dans le répertoire **<Documents>\Altova\MapForce2024\MapForceExamples**.



Comme illustré ci-dessus, la fonction **is-xsi-nil** vérifie si l'attribut `xsi:nil` est "true" pour l'item **state** dans le fichier de source. Si cet attribut est "false", le filtre copiera l'élément parent **Address** dans la cible. Le fichier XML de source ressemble à l'exemple suivant (sauf les déclarations XML et d'espace de noms) :

```
<BranchOffices>
   <Name>Nanonull</Name>
   <Office>
      <Name>Nanonull Research Outpost</Name>
      <EMail>sp@nanonull.com</EMail>
      <Fax xsi:nil="true"/>
      <Phone>+8817 3141 5926</Phone>
      <Address>
         <city>South Pole</city>
         <state xsi:nil="true"/>
         <street xsi:nil="true"/>
         <zip xsi:nil="true"/>
      </Address>
      <Contact>
         <first>Scott</first>
         <last>Amundsen</last>
      </Contact>
   </Office>
</BranchOffices>
```

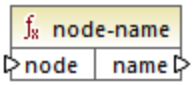
Le résultat du mappage est qu'aucun **Address** n'est copié dans la cible, parce qu'il n'y a qu'un seul **Address** dans la source, et l'attribut `xsi:nil` est défini sur "true" pour l'élément **state**. Par conséquent, la sortie de mappage est le suivant :

```
<BranchOffices>
   <Name>Nanonull</Name>
   <Office>
      <Name>Nanonull Research Outpost</Name>
      <EMail xsi:nil="true"/>
      <Fax>n/a</Fax>
      <Phone>+8817 3141 5926</Phone>
      <Contact>
         <first>Scott</first>
         <last>Amundsen</last>
      </Contact>
   </Office>
</BranchOffices>
```

6.6.7.2 node-name

Retourne le nom qualifié (QName) du nœud connecté. Si le nœud est un nœud XML **text()** un QName vide sera retourné. Cette fonction ne marche que sur les nœuds qui ont un nom. Si XSLT est le langue cible (qui appelle `fn:node-name`), la fonction retourne une séquence vide pour les nœuds qui n'ont pas de nom

Note : Obtenir le nom de nœud n'est pas pris en charge pour les nœuds "File input", les tables de base de données ou les champs, XBRL, Excel, JSON, ou Protocol Buffers.



Langages

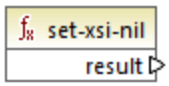
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Argument	Description
node	Paramètre obligatoire. Connecter cette entrée dans le nœud dont vous souhaitez obtenir le nom.

6.6.7.3 set-xsi-nil

Défini le nœud de cible sur xsi:nil.



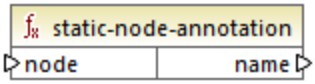
Langages

Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

6.6.7.4 static-node-annotation

Retourne le string avec l'annotation du nœud connecté. L'entrée doit être : (i) un composant de source, ou (ii) une fonction définie par l'utilisateur de type "inline[204]" qui est directement connectée à un paramètre[209] qui est à son tour directement connecté à un nœud dans le mappage d'appel.

La connexion doit être directe. Elle ne peut pas passer à travers un filtre ou une fonction définie par l'utilisateur régulière (pas "inline"). Il s'agit d'une pseudo-fonction, qui est remplacée au moment de la génération par le texte obtenu depuis le nœud connecté, et est donc disponible pour tous les langages



Langages

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

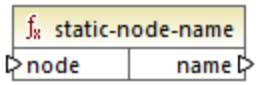
Paramètres

Argument	Description
node	Paramètre obligatoire. Connecter cette entrée vers le nœud dont vous souhaitez obtenir l'annotation.

6.6.7.5 static-node-name

Retourne le string avec le nom du nœud connecté. L'entrée doit être : (i) un composant de source, ou (ii) une fonction définie par l'utilisateur de type "inline[204]" qui est directement connectée à un paramètre[209] qui est, à son tour, directement connecté à un nœud dans le mappage d'appel.

La connexion doit être directe. Elle ne peut pas passer par un filtre ou une fonction non-inlined définie par l'utilisateur. Il s'agit d'une pseudo-fonction, qui est remplacée au moment de la génération par le texte obtenu depuis le nœud connecté, et est donc disponible pour tous les langages.



Langages

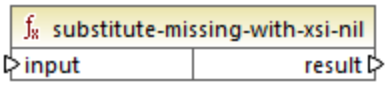
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Paramètres

Argument	Description
node	Paramètre obligatoire. Connecter cette entrée dans le nœud dont vous souhaitez obtenir le nom.

6.6.7.6 substitute-missing-with-xsi-nil

Pour des nœuds avec un contenu simple, cette fonction remplace toute valeur manquante (ou nulle) du composant de source, avec l'attribut `xsi:nil` dans le nœud cible.



Langages

Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

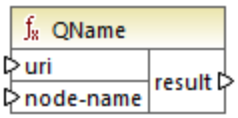
Argument	Description
input	Paramètre obligatoire. Connecter cette entrée dans le nœud dont vous souhaitez obtenir le nom.

6.6.8 core | QName functions

Les fonctions QName permettent de manipuler les Qualified Names (QName) dans les documents XML.

6.6.8.1 QName

Construit un QName depuis un URI d'espace de noms et une partie locale. Utiliser cette fonction pour créer un QName dans un composant de cible. Les paramètres **uri** et **node-name** peuvent être fournis par une fonction constante.



Langages

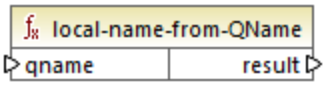
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Description
uri	Obligatoire. Fournit l'URI.
node-name	Obligatoire. Fournit le nom du nœud.

6.6.8.2 local-name-from-QName

Extrait le nom local d'une valeur de type `xs:QName`. Veuillez noter que, contrairement à la fonction local-name qui retourne le nom local du nœud de *nœud*, cette fonction traite le *contenu* de l'item connecté à l'entrée **qname**.



Langages

Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Description
qname	Obligatoire. Fournit la valeur d'entrée de la fonction, de type `xs:QName`.

6.6.8.3 namespace-uri-from-QName

Retourne la partie d'espace de noms d'URI de la valeur QName fournie en tant qu'argument.

Langages

Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

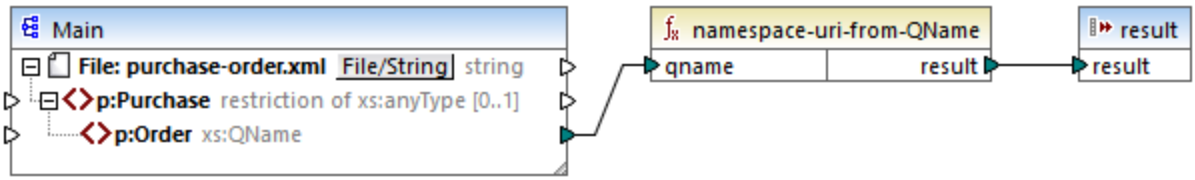
Nom	Description
qname	Obligatoire. Fournit la valeur d'entrée de la fonction.

Exemple

Le fichier XML suivant contient une valeur QName, **o:name**. Veuillez noter que le préfixe "o" est mappé à l'espace de noms **`http://NamespaceTest.com/Order`**.

```
<?xml version="1.0" encoding="utf-8"?>
<p:Purchase xsi:schemaLocation="http://NamespaceTest.com/Purchase Main.xsd"
          xmlns:p="http://NamespaceTest.com/Purchase"
          xmlns:o="http://NamespaceTest.com/Order"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
          <p:Order>o:name</p:Order>
</p:Purchase>
```

Un mappage qui traite la valeur QName et obtient l'URI d'espace de noms est illustrée ci-dessous :



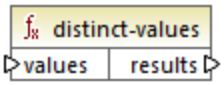
La sortie de ce mappage est **`http://NamespaceTest.com/Order`**.

6.6.9 core | sequence functions

Les fonctions de séquence permettent le traitement des sequences[406] d'entrée et du regroupement de leur contenu.

6.6.9.1 distinct-values

Traite la séquence des valeurs connectées aux entrées **values** et retourne uniquement les valeurs distinctes, en tant que séquence. Cela est utile lorsque vous devez supprimer des valeurs doubles d'une séquence et ne copier que les items uniques dans le composant cible.



Langages

Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

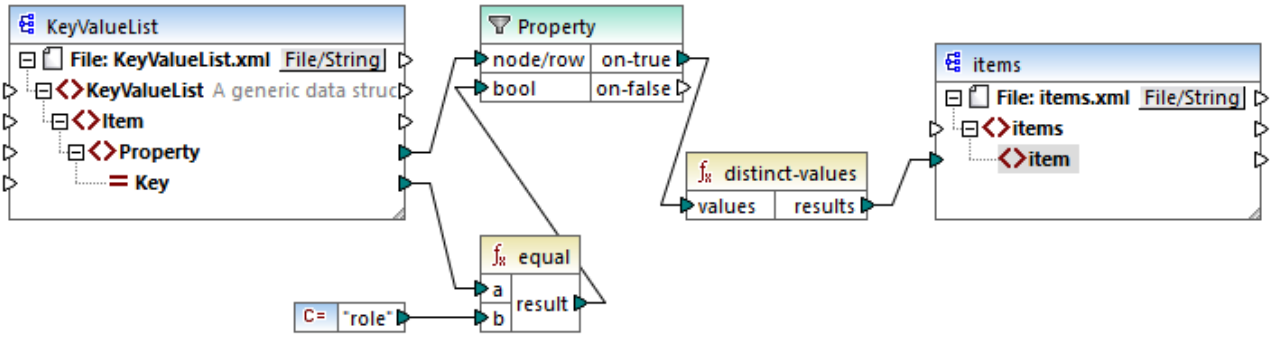
Nom	Description
valeurs	Cette entrée doit recevoir une connexion depuis un item de mappage qui fournit une séquence 406 de zéros ou plus de valeurs. Par exemple, la connexion peut provenir d'un item XML de source .

Exemple

Le fichier XML suivant contient des informations concernant les employés d'une entreprise fictive. Certains employés ont le même rôle ; c'est pourquoi le rôle d'attribut "role" contient des valeurs doubles. Par exemple, "Loby Matise" et "Susi Sanna" ont toutes deux le rôle "Support".

```
<?xml version="1.0" encoding="UTF-8"?>
<KeyValueList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="KeyValueList.xsd">
   <Item>
      <Property Key="role">Manager</Property>
      <Property Key="First">Vernon</Property>
      <Property Key="Last">Callaby</Property>
   </Item>
   <Item>
      <Property Key="role">Programmer</Property>
      <Property Key="First">Frank</Property>
      <Property Key="Last">Further</Property>
   </Item>
   <Item>
      <Property Key="role">Support</Property>
      <Property Key="First">Loby</Property>
      <Property Key="Last">Matise</Property>
   </Item>
   <Item>
      <Property Key="role">Support</Property>
      <Property Key="First">Susi</Property>
      <Property Key="Last">Sanna</Property>
   </Item>
</KeyValueList>
```

Supposons que vous souhaitez extraire une liste de tous les noms de rôle *unique* qui se produisent dans ce fichier XML. Cela peut être obtenu à l'aide d'un mappage comme celui ci-dessous :



Dans le mappage ci-dessus, les choses suivantes se produisent :

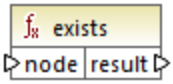
- Chaque élément **Property** provenant du fichier XML de source est traité par un filtre.
- La connexion à l'entrée **bool** du filtre assure que seuls des éléments **Property** où l'attribut **Key** est égale à "role" sont fournis dans le composant de la cible. Le string "role" est fourni par une constante. Veuillez noter que la sortie du filtre produit encore des doubles à ce niveau (puisqu'il y a deux propriétés "Support" qui remplissent la condition du filtre).
- La séquence produite par le filtre est traitée par la fonction **`distinct-values`** qui exclut toute valeur double.

Par conséquent, la sortie de mappage est la suivante (à l'exception des déclarations XML et de schéma) :

```
<items>
   <item>Manager</item>
   <item>Programmer</item>
   <item>Support</item>
</items>
```

6.6.9.2 exists

Retourne **true** si le nœud connecté existe; **false** sinon. Puisqu'elle retourne une valeur booléenne, cette fonction est généralement utilisée avec filters[176], pour filtrer uniquement les enregistrements qui ont (ou peut-être qui n'ont pas) un élément ou un attribut enfant.



Langages

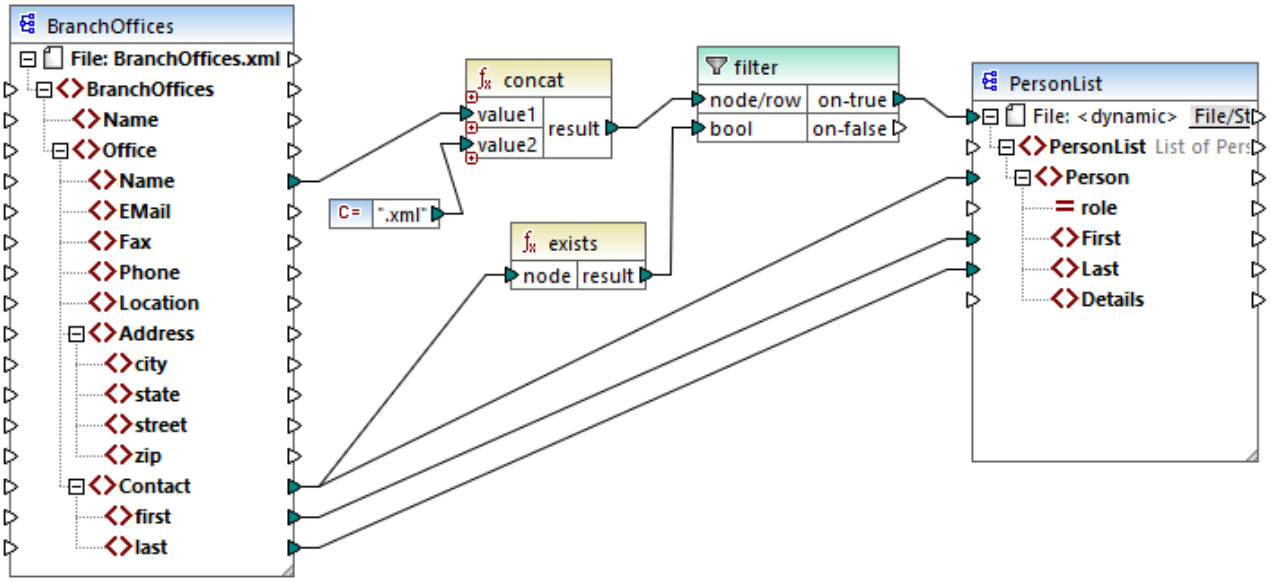
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Description
node	L'existence du nœud doit être testée.

Exemples

Le mappage suivant illustre comment filtrer des données avec l'aide de la fonction **exists**. Ce mappage est appelé **PersonListsForAllBranchOffices.mfd** et peut être trouvé dans le répertoire **<Documents>\Altova\MapForce2024\MapForceExamples**.



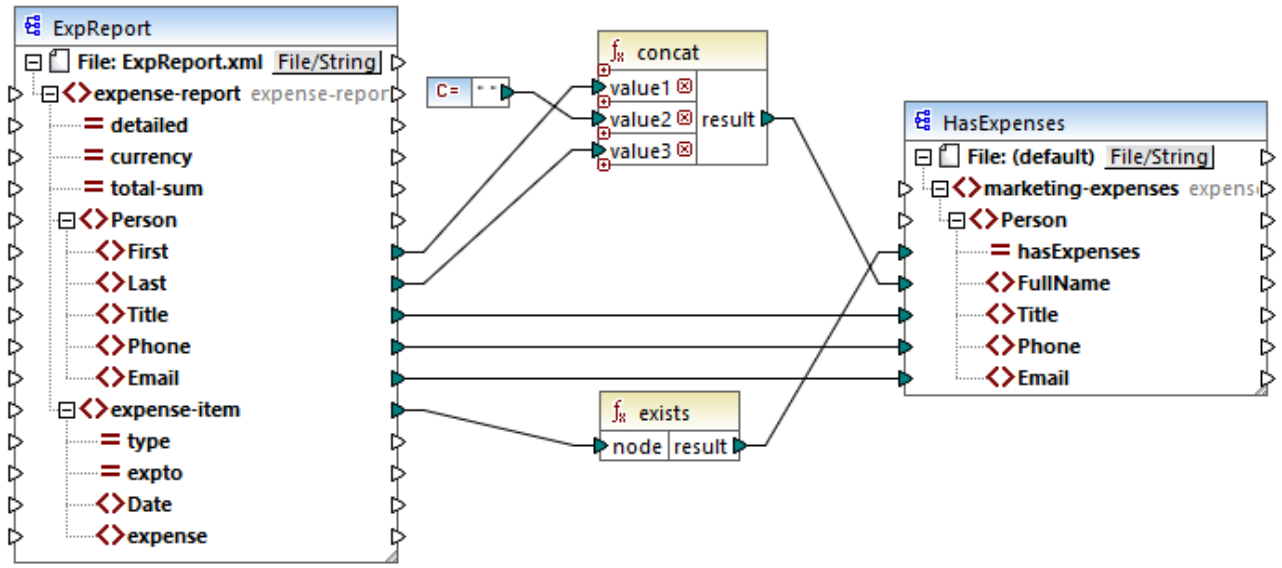
PersonListsForAllBranchOffices.mfd

Dans le fichier source **BranchOffices.xml**, vous trouverez trois éléments **Office**. Notablement, un des trois bureaux n'a pas d'éléments enfant **Contact**. L'objectif de ce mappage est multiple :

a) pour chaque bureau, extraire une liste des contacts qui existent dans ce bureau
b) pour chaque bureau, créer un fichier XML séparer avec le même nom dans le bureau
c) ne pas générer le fichier XML si le bureau n'a pas de contacts.

Pour atteindre ces objectifs, un filtre a été ajouté au mappage. Le filtre fait passer vers la cible uniquement les items **Office** où au moins un item **Contact** existe. Cette condition booléenne est fournie par la fonction **exists**. Si le résultat de la fonction est true, le nom du bureau est concaténé avec le string **.xml** afin de produire le nom de fichier de cible. Pour plus d'informations concernant la génération des noms de fichier depuis le mappage, voir Traiter plusieurs fichiers d'entrée ou de sortie dynamiquement 400 .

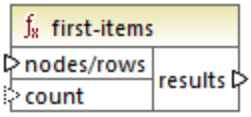
Un autre exemple est le mappage suivant : **<Documents>\Altova\MapForce2024\MapForceExamples\HasMarketingExpenses.mfd**. Ici, si un **expense-item** existe dans le XML source, alors l'attribut **hasExpenses** est défini sur **true** dans le fichier XML de cible.



HasMarketingExpenses.mfd

6.6.9.3 first-items

Retourne les premiers items *N* de la séquence d'entrée, où *N* est fourni par le paramètre **count**.



Langages

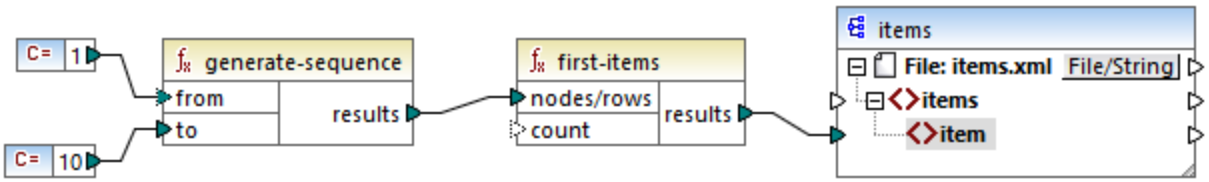
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Description
nodes/rows	Cette entrée doit recevoir une connexion depuis un item de mappage qui fournit une séquence[406] de zéros ou plus de valeurs. Par exemple, la connexion peut provenir d'un item XML de source .
count	Paramètre optionnel. Spécifie combien d'items doivent être extraits depuis la séquence d'entrée. La valeur par défaut est de **1**.

Exemple

Le mappage fictif suivant génère une séquence de 10 valeurs. La séquence est traitée par la fonction **first-items** et le résultat est écrit dans un fichier XML de cible.



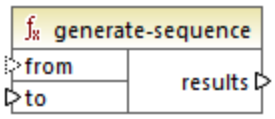
Étant donné que l'argument **count** n'a pas de valeur, la valeur par défaut de **1** s'applique. Par conséquence, seule la première valeur provenant de la séquence est générée dans la sortie du mappage :

```
<items>
   <item>1</item>
</items>
```

Pour un exemple plus réaliste, voir le mappage **FindHighestTemperatures.mfd** discuté dans Fournir les paramètres au mappage 147.

6.6.9.4 generate-sequence

Crée une séquence des entiers en utilisant les paramètres "from" et "to" en tant que les limites.



Langages

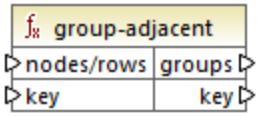
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Description
from	Paramètre optionnel. Spécifie l'entier avec lequel la séquence doit démarrer (bord inférieur). La valeur par défaut est de **1**.
to	Paramètre obligatoire. Spécifie l'entier avec lequel la séquence doit se terminer (bord supérieur).

6.6.9.5 group-adjacent

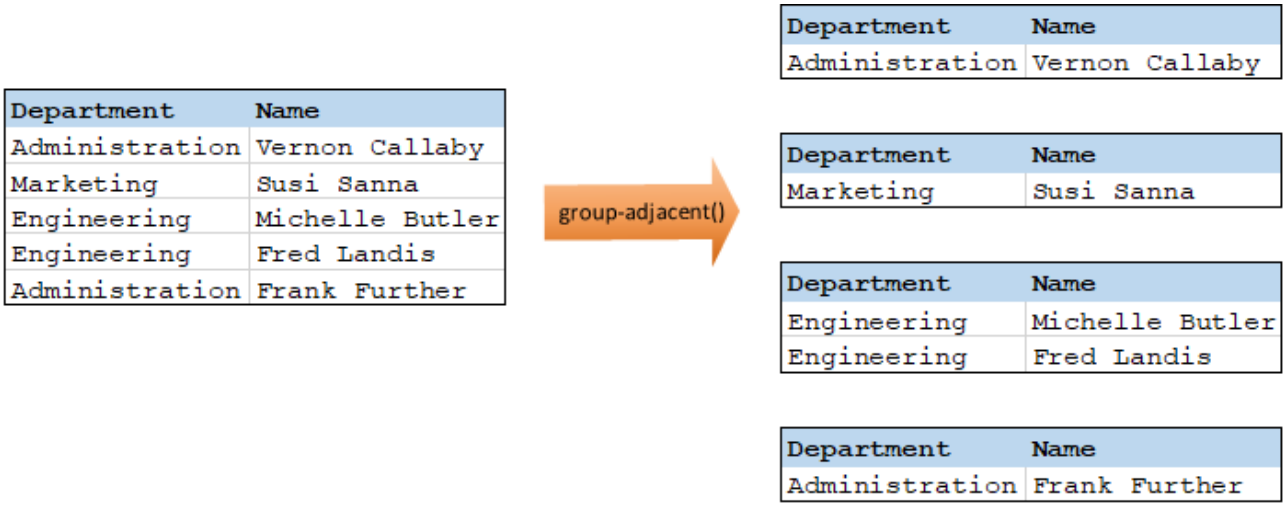
La fonction **group-adjacent** regroupe les items connectés aux entrées **nodes/rows** par la clé connectée à l'entrée **key**. Veuillez noter que cette fonction place les items qui partagent la même clé dans des groupes séparés s'ils ne sont pas adjacents. Si plusieurs items consécutifs (adjacents) partagent la même clé, ils sont placés dans le même groupe.



Par exemple, dans la transformation abstraite illustrée ci-dessous, la clé de regroupement est "Department". Le côté gauche du diagramme montre les données d'entrée tandis que le côté droit montre les données de sortie après le regroupement. Les événements suivants se produisent lorsque la transformation est exécutée :

- Tout d'abord, la première clé, "Administration", crée un nouveau groupe.
- La clé suivante est différente, donc un deuxième groupe est créé :"Marketing".
- La troisième clé est aussi différente, donc un groupe supplémentaire est créé : "Engineering".
- La quatrième clé est la même que la troisième, c'est pourquoi cet enregistrement est placé dans le groupe déjà existant.
- Enfin, la cinquième clé est différente de la quatrième et cela crée le dernier groupe.

Comme illustré ci-dessous, "Michelle Butler" et "Fred Landis" ont été regroupés car ils partagent la même clé et sont adjacents. Néanmoins, "Vernon Callaby" et "Frank Further" se trouvent dans des groupes séparés étant donné qu'ils ne sont pas adjacents, même s'ils possèdent la même clé.



Langages

Built-in, C++, C#, Java, XSLT 2.0, XSLT 3.0.

Paramètres

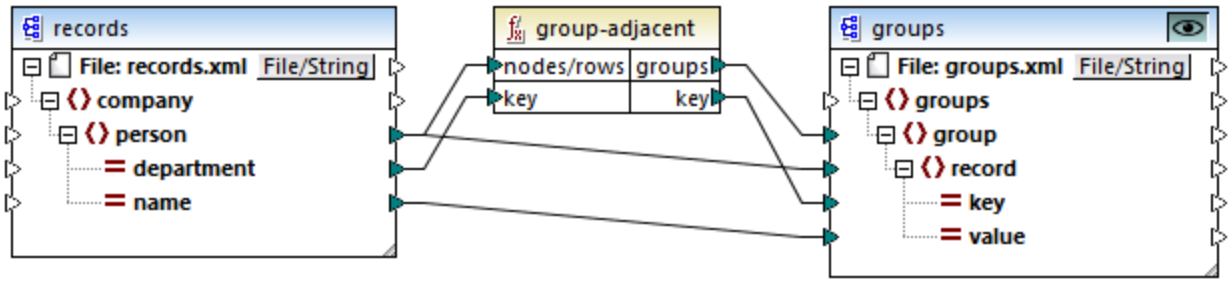
Nom	Description
nodes/rows	Cette entrée doit recevoir une connexion depuis un item de mappage qui fournit une séquence 406 de zéros ou plus de valeurs. Par exemple, la connexion peut provenir d'un item XML de source .
key	La clé avec laquelle regrouper des items.

Exemple

Partons du principe que vos données de source existent sous la forme d'un fichier XML contenant les éléments suivants (veuillez noter que dans l'extrait de code suivant, l'espace de noms et des déclarations XML ont été supprimés pour plus de simplicité).

```
<company>
   <person department="Administration" name="Vernon Callaby"/>
   <person department="Marketing" name="Susi Sanna"/>
   <person department="Engineering" name="Michelle Butler"/>
   <person department="Engineering" name="Fred Landis"/>
   <person department="Administration" name="Frank Further"/>
</company>
```

L'exigence commerciale est de grouper des enregistrements de personnes par département, s'ils sont adjacents. Pour ce faire, le mappage suivant invoque la fonction **group-adjacent** et fournit **department** en tant que **key**.



Le résultat de mappage est le suivant :

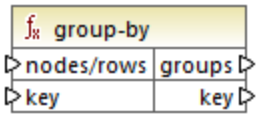
```
<groups>
   <group>
      <record key="Administration" value="Vernon Callaby"/>
   </group>
   <group>
      <record key="Marketing" value="Susi Sanna"/>
   </group>
   <group>
      <record key="Engineering" value="Michelle Butler"/>
```

```
      <record key="Engineering" value="Fred Landis"/>
   </group>
   <group>
      <record key="Administration" value="Frank Further"/>
   </group>
</groups>
```

Cet exemple, avec d'autres exemples de regroupement, fait partie du fichier de mappage suivant : **<Documents>\Altova\MapForce2024\MapForceExamples\Tutorial\GroupingFunctions.mfd**. Ne pas oublier de cliquer sur la touche **Aperçu** applicable à la fonction que vous souhaitez consulter préalablement, avant de cliquer sur l'onglet **Sortie**.

6.6.9.6 group-by

La fonction **group-by** crée des groupes d'enregistrements conformément à certaines clés de regroupement que vous aurez spécifiées.



Par exemple, dans la transformation abstraite illustrée ci-dessous, la clé de regroupement est "Department". Étant donné qu'il y a trois départements au total, l'application de la fonction **group-by** créera trois groupes :

Department	Name
Administration	Vernon Callaby
Marketing	Susi Sanna
Engineering	Michelle Butler
Engineering	Fred Landis
Administration	Frank Further

group-by()

Department	Name
Administration	Vernon Callaby
Administration	Frank Further

Department	Name
Marketing	Susi Sanna

Department	Name
Engineering	Michelle Butler
Engineering	Fred Landis

Langages

Built-in, C++, C#, Java, XSLT 2.0, XSLT 3.0.

Paramètres

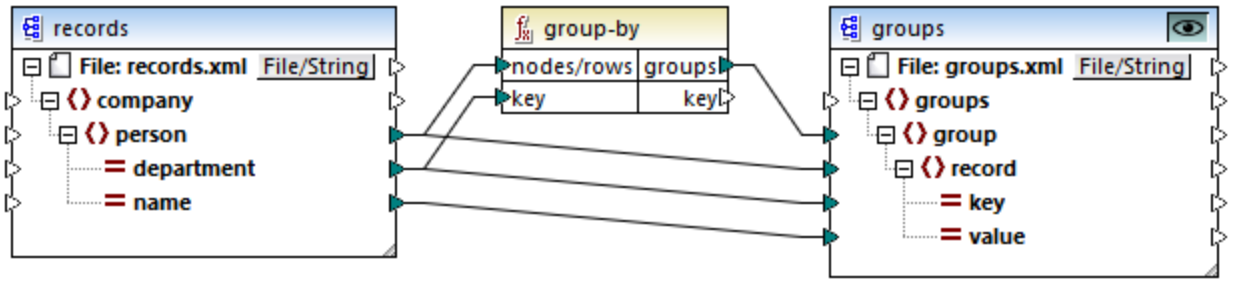
Nom	Description
nodes/rows	Cette entrée doit recevoir une connexion depuis un item de mappage qui fournit une séquence 406 de zéros ou plus de valeurs. Par exemple, la connexion peut provenir d'un item XML de source .
key	La clé avec laquelle regrouper des items.

Exemple 1

Partons du principe que vos données de source existent sous la forme d'un fichier XML contenant les éléments suivants (veuillez noter que dans l'extrait de code suivant, l'espace de noms et des déclarations XML ont été supprimés pour plus de simplicité).

```
<company>
   <person department="Administration" name="Vernon Callaby"/>
   <person department="Marketing" name="Susi Sanna"/>
   <person department="Engineering" name="Michelle Butler"/>
   <person department="Engineering" name="Fred Landis"/>
   <person department="Administration" name="Frank Further"/>
</company>
```

L'exigence commerciale est de regrouper des enregistrements de personnes par département. Pour ce faire, le mappage suivant invoque la fonction **group-by** et fournit **department** en tant que clé.



Le résultat de mappage est le suivant :

```
<groups>
   <group>
      <record key="Administration" value="Vernon Callaby"/>
      <record key="Administration" value="Frank Further"/>
   </group>
   <group>
      <record key="Marketing" value="Susi Sanna"/>
   </group>
   <group>
      <record key="Engineering" value="Michelle Butler"/>
```

```
      <record key="Engineering" value="Fred Landis"/>
   </group>
</groups>
```

Cet exemple, avec d'autres exemples de regroupement, fait partie du fichier de mappage suivant : **<Documents>\Altova\MapForce2024\MapForceExamples\Tutorial\GroupingFunctions.mfd**. Ne pas oublier de cliquer sur la touche **Aperçu**  applicable à la fonction que vous souhaitez consulter préalablement, avant de cliquer sur le volet **Sortie**.

Exemple 2

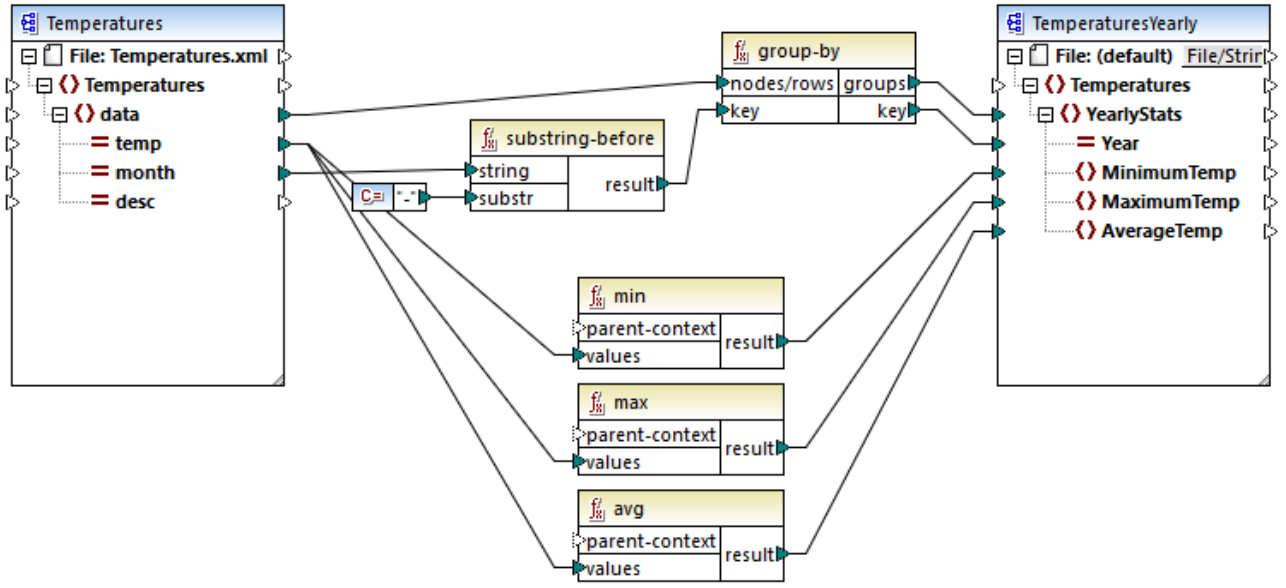
Cet exemple vous montre comment regrouper des enregistrements avec l'aide de la fonction **group-by**, de même, il illustre comment agréger des données. Cet exemple est accompagné par un mappage de démonstration disponible sous le chemin suivant : **<Documents>\Altova\MapForce2024\MapForceExamples\GroupTemperaturesByYear.mfd**. Ce mappage lit les données provenant d'un fichier XML qui contient un journal de températures mensuelles, comme illustré dans le code ci-dessous :

```
<Temperatures>
   <data temp="-3.6" month="2006-01" />
   <data temp="-0.7" month="2006-02" />
   <data temp="7.5" month="2006-03" />
   <data temp="12.4" month="2006-04" />
   <data temp="16.2" month="2006-05" />
   <data temp="19" month="2006-06" />
   <data temp="22.7" month="2006-07" />
   <data temp="23.2" month="2006-08" />
   <data temp="18.7" month="2006-09" />
   <data temp="11.2" month="2006-10" />
   <data temp="9.1" month="2006-11" />
   <data temp="0.8" month="2006-12" />
   <data temp="-3.2" month="2007-01" />
   <data temp="-0.3" month="2007-02" />
   <data temp="6.5" month="2007-03" />
   <data temp="10.6" month="2007-04" />
   <data temp="19" month="2007-05" />
   <data temp="20.3" month="2007-06" />
   <data temp="22.3" month="2007-07" />
   <data temp="20.7" month="2007-08" />
   <data temp="19.2" month="2007-09" />
   <data temp="12.9" month="2007-10" />
   <data temp="8.1" month="2007-11" />
   <data temp="1.9" month="2007-12" />
</Temperatures>
```

Les exigences commerciales de ce mappage sont doubles :

1. Regrouper les températures de chaque année.
2. Trouver le minimum, le maximum, et la température moyenne de chaque année.

Afin d'atteindre le premier objectif, nous utilisons la fonction **group-by**. Afin d'atteindre le second objectif, nous utilisons les fonctions d'agrégation **min**[238], **max**[237], et **avg**[235].



GroupTemperaturesByYear.mfd

Pour exécuter un mappage, MapForce (et il s'agit là de la méthode recommandée pour commencer à lire un mappage) consulter l'item supérieur du composant de cible. Dans cet exemple, un item **YearlyStats** sera créé pour chaque groupe retourné par la fonction **group-by**. La fonction **group-by** prend en tant que premier argument tous les items **data** depuis la source et les regroupe selon ce qui est connecté à l'entrée **key**. Puisque l'exigence est de regrouper des températures par année, il faut tout d'abord obtenir l'année. Pour ce faire, la fonction **substring-before**[312] extrait la partie de l'année depuis l'attribut **month** de chaque élément **data**. Concrètement, il prend en tant qu'argument la valeur de **month** et retourne la partie avant la première apparition de **substr**. Comme illustré ci-dessus, dans cet exemple, **substr** est défini pour le caractère de tiret ; ainsi, pour une valeur "2006-01", la fonction retournera "2006".

Enfin, les valeurs de **MinimumTemp**, **MaximumTemp** et **AverageTemp** sont obtenues en connectant ces items avec les fonctions d'agrégat respectives : **min**, **max** et **avg**. Les trois fonctions prennent en tant qu'entrée la séquence des températures lues depuis le composant de source. Ces fonctions ne nécessitent pas d'argument **parent-context**, parce qu'elles fonctionnent déjà dans le contexte de chaque groupe. En d'autres mots, il existe une connexion parent, de **data** à **YearlyStats** qui fournit le contexte pour chaque fonction d'agrégation sur laquelle vous souhaitez travailler.

Pour consulter la sortie de mappage, cliquez sur le volet **Sortie**. Veuillez noter que le nombre de groupes doit coïncider avec le nombre d'années obtenues en lisant le fichier de source, par exemple :

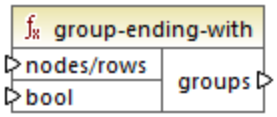
```
<Temperatures>
   <YearlyStats Year="2006">
      <MinimumTemp>-3.6</MinimumTemp>
      <MaximumTemp>23.2</MaximumTemp>
      <AverageTemp>11.375</AverageTemp>
   </YearlyStats>
   <YearlyStats Year="2007">
```

```
      <MinimumTemp>-3.2</MinimumTemp>
      <MaximumTemp>22.3</MaximumTemp>
      <AverageTemp>11.5</AverageTemp>
   </YearlyStats>
</Temperatures>
```

Note : Pour une plus grande simplicité, les listes de code ci-dessus contiennent moins de données que l'entrée et la sortie véritables utilisés dans le mappage de démonstration.

6.6.9.7 group-ending-with

La fonction **group-ending-with** prend une condition booléenne en tant qu'argument. Si la condition booléenne est vraie, un nouveau groupe est créé, terminant avec l'enregistrement qui satisfait la condition.



Dans l'exemple ci-dessous, la condition est que "Key" doit être égal à "trailing". Cette condition est vraie pour les troisième et quatrièmes enregistrements, en résultat, deux groupes sont créés :

Key	Value
line	A
line	B
trailing	Total 1
line	C
trailing	Total 2

group-ending-with()

Key	Value
line	A
line	B
trailing	Total 1

Key	Value
line	C
trailing	Total 2

Note : Un groupe supplémentaire est créé si des enregistrements existent après le dernier qui satisfait à la condition. Par exemple, si il existait plus d'enregistrements "line" après le dernier enregistrement "trailing", ceux-ci seraient tous placés dans un nouveau groupe.

Langages

Built-in, C++, C#, Java, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Description
nodes/rows	Cette entrée doit recevoir une connexion depuis un item de mappage qui fournit une séquence 406 de zéros ou plus de valeurs. Par exemple, la

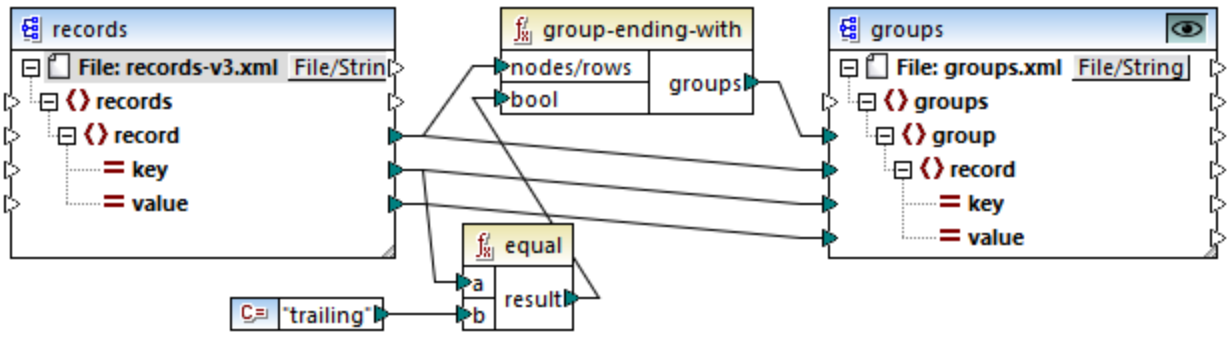
Nom	Description
	connexion peut provenir d'un item XML de source .
bool	Fournit la condition Booléenne qui lance un nouveau groupe si **true**.

Exemple

Partons du principe que vos données de source existent sous la forme d'un fichier XML contenant les éléments suivants (veuillez noter que dans l'extrait de code suivant, l'espace de noms et des déclarations XML ont été supprimés pour plus de simplicité).

```
<records>
   <record key="line" value="A"/>
   <record key="line" value="B"/>
   <record key="trailing" value="Total 1"/>
   <record key="line" value="C"/>
   <record key="trailing" value="Total 2"/>
</records>
```

L'exigence commerciale est de créer des groupes pour chaque enregistrement "trailing" Chaque groupe doit aussi inclure des enregistrements "line" qui précède l'enregistrement "trailing". Pour ce faire, le mappage suivant invoque la fonction **group-ending-with**. Dans le mappage ci-dessous, à chaque fois que le nom **key** est égal à "trailing", l'argument fournit en **bool** devient **true**, et un nouveau groupe est créé.



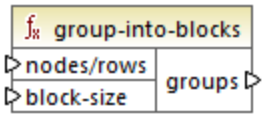
Le résultat de mappage est le suivant :

```
<groups>
   <group>
      <record key="line" value="A"/>
      <record key="line" value="B"/>
      <record key="trailing" value="Total 1"/>
   </group>
   <group>
      <record key="line" value="C"/>
      <record key="trailing" value="Total 2"/>
   </group>
</groups>
```

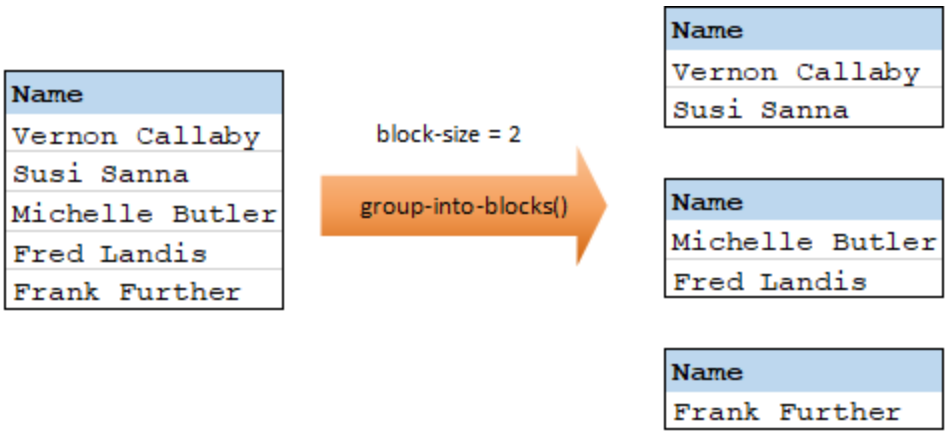
Cet exemple, avec d'autres exemples de regroupement, fait partie du fichier de mappage suivant : **<Documents>\Altova\MapForce2024\MapForceExamples\Tutorial\GroupingFunctions.mfd**. Ne pas oublier de cliquer sur la touche **Aperçu** applicable à la fonction que vous souhaitez consulter préalablement, avant de cliquer sur l'onglet **Sortie**.

6.6.9.8 group-into-blocks

La fonction **`group-into-blocks`** crée des groupes égaux qui contiennent exactement N items, et où N est la valeur que vous fournissez à l'argument `block-size`. Veuillez noter que le dernier groupe peut contenir N items ou moins, selon le nombre d'items se trouvant dans la source.



Dans l'exemple ci-dessous, `block-size` est 2. Étant donné qu'il existe au total cinq items, chaque groupe contient exactement deux items, sauf pour le dernier.



Langages

Built-in, C++, C#, Java, XSLT 2.0, XSLT 3.0.

Paramètres

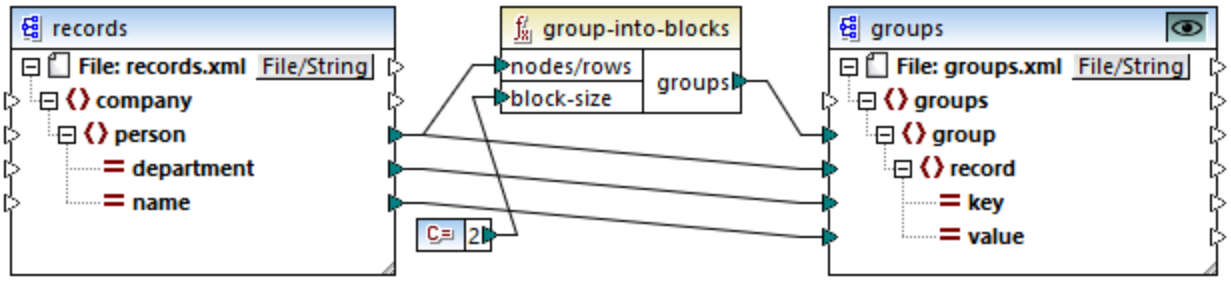
Nom	Description
nodes/rows	Cette entrée doit recevoir une connexion depuis un item de mappage qui fournit une séquence 406 de zéros ou plus de valeurs. Par exemple, la connexion peut provenir d'un item XML de source .
block-size	Spécifie la taille de chaque groupe

Exemple

Partons du principe que vos données de source existent sous la forme d'un fichier XML contenant les éléments suivants (veuillez noter que dans l'extrait de code suivant, l'espace de noms et des déclarations XML ont été supprimés pour plus de simplicité).

```
<company>
   <person department="Administration" name="Vernon Callaby"/>
   <person department="Marketing" name="Susi Sanna"/>
   <person department="Engineering" name="Michelle Butler"/>
   <person department="Engineering" name="Fred Landis"/>
   <person department="Administration" name="Frank Further"/>
</company>
```

L'exigence commerciale est de grouper des enregistrements de personnes en deux items chacun. Pour ce faire, le mappage suivant invoque la fonction **group-into-blocks** et fournit la valeur d'entier "2" en tant que **block-size**.



Le résultat de mappage est le suivant :

```
<groups>
   <group>
      <record key="Administration" value="Vernon Callaby"/>
      <record key="Marketing" value="Susi Sanna"/>
   </group>
   <group>
      <record key="Engineering" value="Michelle Butler"/>
      <record key="Engineering" value="Fred Landis"/>
   </group>
   <group>
      <record key="Administration" value="Frank Further"/>
   </group>
</groups>
```

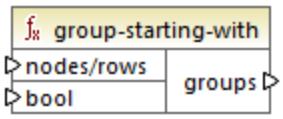
Veuillez noter que le dernier groupe ne contient qu'un seul item, puisque le nombre total d'items (5) ne peut pas être divisé par 2.

Cet exemple, avec d'autres exemples de regroupement, fait partie du fichier de mappage suivant : **<Documents>\Altova\MapForce2024\MapForceExamples\Tutorial\GroupingFunctions.mfd**. Ne pas

oublier de cliquer sur la touche **Aperçu** applicable à la fonction que vous souhaitez consulter préalablement, avant de cliquer sur l'onglet **Sortie**.

6.6.9.9 group-starting-with

La fonction **group-starting-with** prend une condition booléenne en tant qu'argument. Si celle-ci est vraie, un nouveau groupe est créé, commençant avec l'enregistrement qui satisfait à la condition.



Dans l'exemple ci-dessous, la condition est que "Key" doit être égal à "heading". Cette condition est vraie pour le premier et le quatrième enregistrement, donc deux groupes sont créés :

Note: Un groupe supplémentaire est créé si des enregistrements existent après le premier qui satisfait à la condition. Par exemple, s'il existait plus d'enregistrements "line" avant le premier enregistrement "heading", ceux-ci seraient tous placés dans un nouveau groupe.

Langages

Built-in, C++, C#, Java, XSLT 2.0, XSLT 3.0.

Paramètres

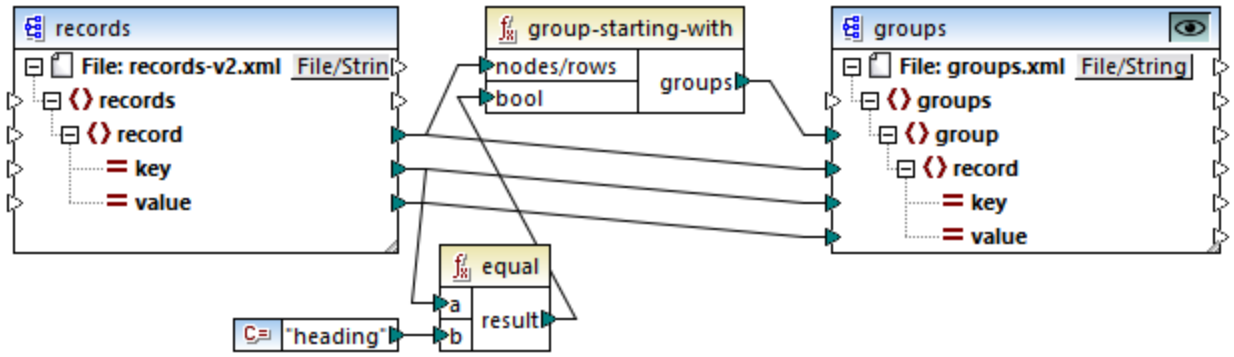
Nom	Description
nodes/rows	Cette entrée doit recevoir une connexion depuis un item de mappage qui fournit une séquence 406 de zéros ou plus de valeurs. Par exemple, la connexion peut provenir d'un item XML de source .
bool	Fournit la condition Booléenne qui lance un nouveau groupe si **true**.

Exemple

Partons du principe que vos données de source existent sous la forme d'un fichier XML contenant les éléments suivants (veuillez noter que dans l'extrait de code suivant, l'espace de noms et des déclarations XML ont été supprimés pour plus de simplicité).

```
<records>
   <record key="heading" value="Intro"/>
   <record key="line" value="A"/>
   <record key="line" value="B"/>
   <record key="heading" value="Body"/>
   <record key="line" value="C"/>
</records>
```

L'exigence commerciale est de créer des groupes pour chaque enregistrement "heading". Chaque groupe doit aussi inclure tout enregistrement "line" qui suit l'enregistrement "heading". Pour ce faire, le mappage suivant invoque la fonction **group-starting-with**. Dans le mappage ci-dessous, dès que le nom **key** est égal à "heading", l'argument fourni en **bool** devient **true** et un nouveau groupe est créé.



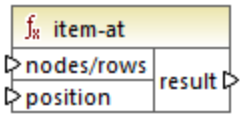
Le résultat de mappage est le suivant :

```
<groups>
   <group>
      <record key="heading" value="Intro"/>
      <record key="line" value="A"/>
      <record key="line" value="B"/>
   </group>
   <group>
      <record key="heading" value="Body"/>
      <record key="line" value="C"/>
   </group>
</groups>
```

Cet exemple, avec d'autres exemples de regroupement, fait partie du fichier de mappage suivant : **<Documents>\Altova\MapForce2024\MapForceExamples\Tutorial\GroupingFunctions.mfd**. Ne pas oublier de cliquer sur la touche **Aperçu** applicable à la fonction que vous souhaitez consulter préalablement, avant de cliquer sur l'onglet **Sortie**.

6.6.9.10 item-at

Retourne un item depuis la séquence de **nodes/rows** fournie en tant qu'argument, à la position fournie par l'argument **position**. Le premier item se trouve à la position **1**.



Langages

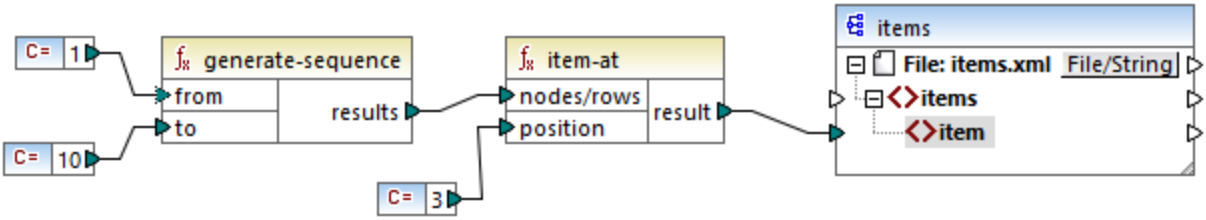
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Description
nodes/rows	Cette entrée doit recevoir une connexion depuis un item de mappage qui fournit une séquence 406 de zéros ou plus de valeurs. Par exemple, la connexion peut provenir d'un item XML de source .
position	Cet entier spécifie quel item provenant de la séquence des items doit être retournée.

Exemple

Le mappage fictif suivant génère une séquence de 10 valeurs. La séquence est traitée par la fonction **item-at** et le résultat est écrit dans un fichier XML de cible.

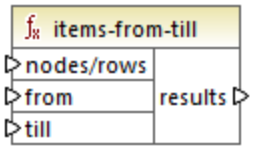


Étant donné que l'argument **position** est défini sur **3**, seule la troisième valeur provenant de la séquence est transmise vers la cible. Par conséquent, la sortie de mappage est la suivante (à l'exception des déclarations XML et de schéma) :

```
<items>
   <item>3</item>
</items>
```

6.6.9.11 items-from-till

Retourne une séquence de **nodes/rows** en utilisant les paramètres "from" et "till" en tant que limites. Le premier item se trouve à la position **1**.



Langages

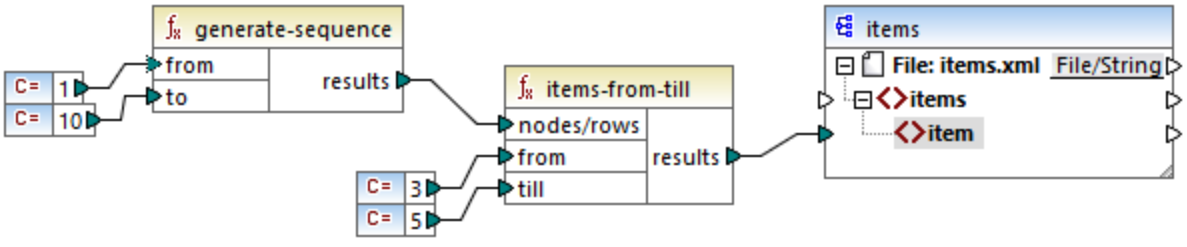
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Description
nodes/rows	Cette entrée doit recevoir une connexion depuis un item de mappage qui fournit une séquence 406 de zéros ou plus de valeurs. Par exemple, la connexion peut provenir d'un item XML de source .
from	Cet entier spécifie la position de démarrage à partir de laquelle les items doivent être extraits.
till	Cet entier spécifie la position jusqu'à laquelle les items doivent être extraits.

Exemple

Le mappage fictif suivant génère une séquence de 10 valeurs. La séquence est traitée par la fonction **item-from-till** et le résultat est écrit dans un fichier XML de cible.



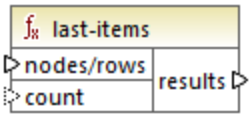
Étant donné que les arguments **from** et **till** sont définis sur **3** et **5**, respectivement, seul le sous-ensemble de valeurs provenant de **3** à **5** est transmis vers la cible. Par conséquent, la sortie de mappage est la suivante (à l'exception des déclarations XML et de schéma) :

```
<items>
   <item>3</item>
```

```
   <item>4</item>
   <item>5</item>
</items>
```

6.6.9.12 last-items

Retourne les premiers items *N* de la séquence d'entrée, où *N* est fourni par le paramètre **count**. Le premier item se trouve à la position "1".



Langages

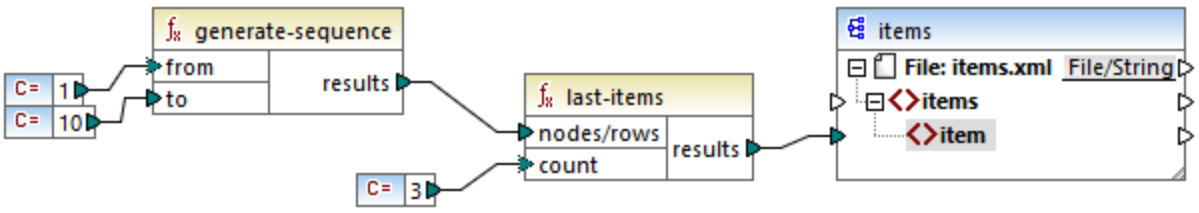
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Description
nodes/rows	Cette entrée doit recevoir une connexion depuis un item de mappage qui fournit une séquence[406] de zéros ou plus de valeurs. Par exemple, la connexion peut provenir d'un item XML de source .
count	Paramètre optionnel. Spécifie combien d'items doivent être extraits depuis la séquence d'entrée. La valeur par défaut est de **1**.

Exemple

Le mappage fictif suivant génère une séquence de 10 valeurs. La séquence est traitée par la fonction **last-items** et le résultat est écrit dans un fichier XML de cible.



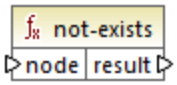
Étant donné que l'argument count est défini sur **3**, seule les trois dernières valeurs provenant de la séquence sont transmises vers la cible. Par conséquent, la sortie de mappage est la suivante (à l'exception des déclarations XML et de schéma) :

```
<items>
```

```
   <item>8</item>
   <item>9</item>
   <item>10</item>
</items>
```

6.6.9.13 not-exists

Retourne **false** si le nœud connecté existe; **true** sinon. Cette fonction est l'inverse de la fonction exists 278 mais à part cela elle a la même utilisation.



Langages

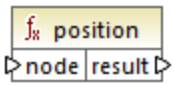
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Description
node	Le nœud à tester pour non-existence.

6.6.9.14 position

Retourne la position d'un item dans le cadre de la séquence des items actuellement en cours de traitement. Cela peut être utilisé, par exemple pour numéroter automatiquement des items séquentiellement.



Langages

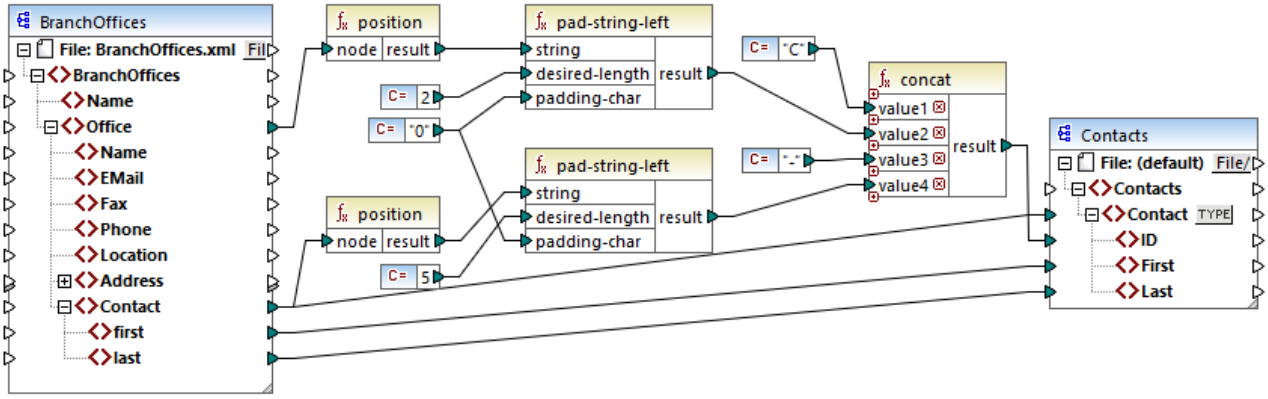
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Description
node	Cette entrée doit recevoir une connexion depuis un item de mappage qui fournit une séquence 406 de zéros ou plus de valeurs. Par exemple, la connexion peut provenir d'un item XML de source .

Exemple

Le mappage suivant illustre l'utlisation de la fonction **position** pour générer des valeurs d'identification uniques dans des données générées par le mappage. Ce mappage est accompagné par un fichier de design de mappage qui est disponible sous le chemin suivant :
<Documents>\Altova\MapForce2024\MapForceExamples\ContactsFromBranchOffices.mfd.



ContactsFromBranchOffices.mfd

Dans le mappage ci-dessus, le fichier XML de source contient trois filiales. Une filiale peut contenir un nombre arbitraire d'items enfant **Contact**. Les objectifs du mappage sont les suivants :

- Extraire tous les items **Contact** du fichier XML de source et les écrire dans le fichier XML de cible.
- Chaque contact doit être attribué à un numéro d'identification unique (l'item **ID** dans le XML de cible).
- L'ID de chaque contact doit prendre la forme de **CXX-YYYYY**, où X identifie le numéro de bureau et Y identifie le numéro de contact. Si le numéro du bureau a moins de deux caractères, il faut rajouter des zéros à gauche. De même, si le numéro de contact a moins de cinq caractères, il faut rajouter des zéros à gauche. Par conséquent, un numéro d'identification valide du premier contact du premier bureau devrait ressembler à l'exemple suivant : **C01-00001**.

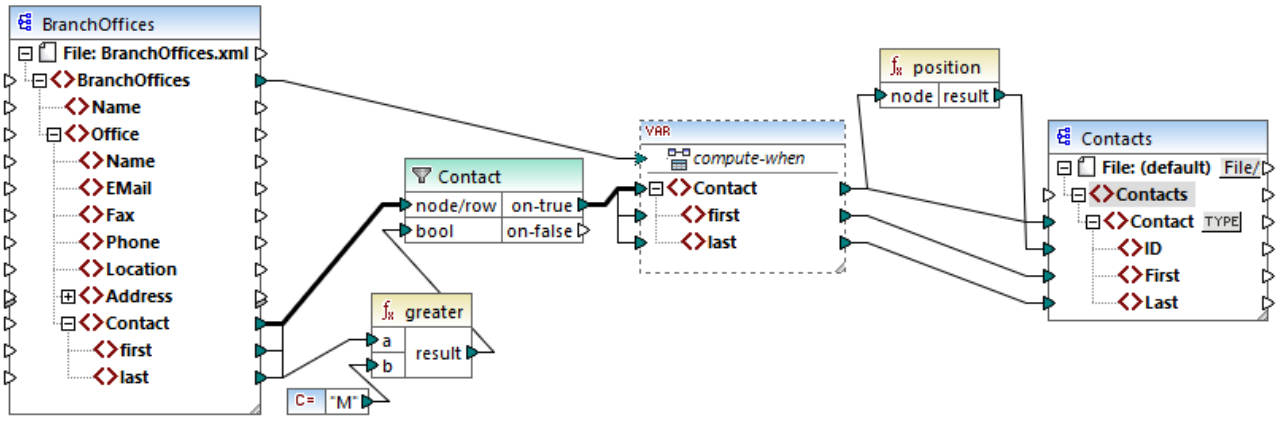
Pour obtenir les objectifs de mappage, plusieurs fonctions MapForce ont été utilisées, y compris la fonction **position**. La fonction supérieure **position** obtient la position de chaque bureau. La fonction inférieure obtient la position de chaque contact, dans le contexte de chaque bureau.

Lorsque vous utilisez la fonction **position**, il est important de considérer le contexte de mappage 407 actuel. Plus spécifiquement, lorsque le mappage est exécuté, le contexte de mappage initial est établi depuis l'item root du composant cible vers l'item de source qui y est connecté (même indirectement via des fonctions). Dans cet exemple, la fonction supérieure **position** traite *la séquence de tous les bureaux* et elle génère initialement la valeur valeur 1, correspondant au premier bureau dans la séquence. La fonction inférieure **position** génère des nombres séquentiels correspondant à la position du contact *dans le contexte de ce bureau* (1, 2, 3, etc.). Veuillez noter que cette séquence "interne" sera réinitialisée (et donc recommencera avec le numéro 1) lorsque le bureau suivant est traité. Les deux fonctions **pad-string-left** appliquent le remplissage aux numéros générés, conformément aux exigences déclarées précédemment. La fonction concat fonctionne *dans le contexte de chaque contact* (en raison de la connexion parent provenant de la source vers la cible **Contact**). Elle rejoint toutes les valeurs calculées et retourne le numéro d'identification unique de chaque contact.

La sortie générée depuis le mappage ci-dessus est affiché ci-dessous (veuillez noter que certains des enregistrements ont été supprimés pour la lecture) :

```
<Contacts>
   <Contact>
      <ID>C01-00001</ID>
      <First>Vernon</First>
      <Last>Callaby</Last>
   </Contact>
   <Contact>
      <ID>C01-00002</ID>
      <First>Frank</First>
      <Last>Further</Last>
   </Contact>
   <!-- ... -->
   <Contact>
      <ID>C02-00001</ID>
      <First>Steve</First>
      <Last>Meier</Last>
   </Contact>
   <Contact>
      <ID>C02-00002</ID>
      <First>Theo</First>
      <Last>Bone</Last>
   </Contact>
   <!-- ... -->
</Contacts>
```

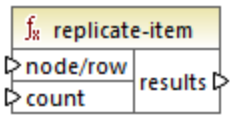
Il peut aussi exister des cas où vous souhaitez obtenir la position d'items résultant après l'application d'un filtre[176]. Veuillez noter que le composant de filtre n'est pas une fonction de séquence et qu'il ne peut pas être utilisé *directement* en conjonction avec la fonction de **position** pour trouver la position des items filtrés. Indirectement, cela est possible en ajoutant un composant de variable[158] dans le mappage. Par exemple, le mappage ci-dessous est une version simplifiée de la version précédente. Son fichier de design de mappage est disponible sous le chemin suivant :
<Documents>\Altova\MapForce2024\MapForceExamples\PositionInFilteredSequence.mfd.



Le résultat des variables dans MapForce est toujours des séquences. C'est pourquoi, dans le mappage ci-dessus, la fonction **position** itère à travers la séquence créée par la variable et retourne la position de chaque item dans cette séquence. Ce mapping est discuté plus en détail dans Exemple : Filtrer et Numéroter des nœuds 166.

6.6.9.15 replicate-item

Répète chaque item dans la séquence d'entrée le nombre de fois spécifié dans l'argument **count**. Si vous connectez un seul item dans l'entrée **node/row**, la fonction retournera *N* items, où *N* est la valeur de l'argument **count**. Si vous connectez une séquence d'item dans l'entrée **node/row**, la fonction répètera chaque item individuel dans la séquence **count** fois, en traitant un item à la fois. Par exemple, si count est **2**, la séquence `1,2,3` produira `1,1,2,2,3,3`. Il est aussi possible de fournir une valeur **count** différente pour chaque item dans la séquence d'entrée, comme illustré dans l'exemple ci-dessous.



Langages

Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Description
node/row	Cette entrée doit recevoir une connexion depuis un item de mappage qui fournit une séquence 406 de zéros ou plus de valeurs. Par exemple, la connexion peut provenir d'un item XML de source .
count	Spécifie le nombre de fois nécessaire pour répliquer chaque item ou séquence connectée à **node/row**.

Exemple

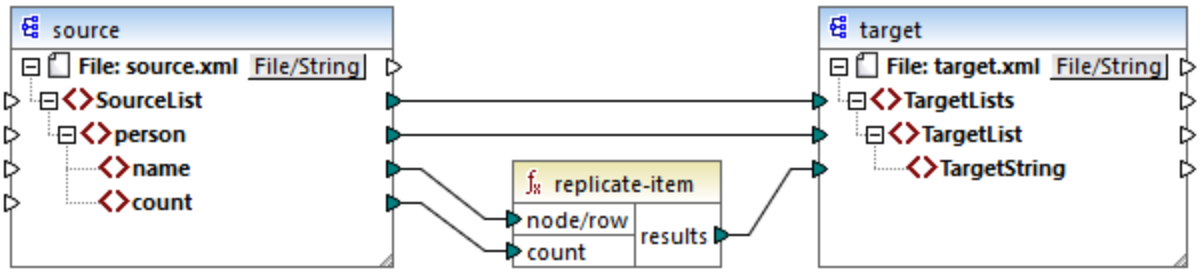
Imaginons que vous avez un fichier XML source comportant la structure suivante :

```
<SourceList>
   <person>
      <name>Michelle</name>
      <count>2</count>
   </person>
   <person>
      <name>Ted</name>
      <count>4</count>
   </person>
   <person>
      <name>Ann</name>
```



```
      <count>3</count>
   </person>
</SourceList>
```

Avec l'aide de la fonction **replicate-item**, vous pouvez répéter chaque nom de personne un nombre différent de fois dans un composant de cible. Pour ce faire, connecter le nœud **count** de chaque personne à l'entrée **count** de la fonction **replicate-item** :

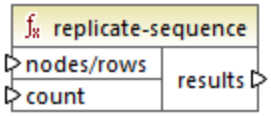


La sortie est la suivante :

```
<TargetLists>
   <TargetList>
      <TargetString>Michelle</TargetString>
      <TargetString>Michelle</TargetString>
   </TargetList>
   <TargetList>
      <TargetString>Ted</TargetString>
      <TargetString>Ted</TargetString>
      <TargetString>Ted</TargetString>
      <TargetString>Ted</TargetString>
   </TargetList>
   <TargetList>
      <TargetString>Ann</TargetString>
      <TargetString>Ann</TargetString>
      <TargetString>Ann</TargetString>
   </TargetList>
</TargetLists>
```

6.6.9.16 replicate-sequence

Répéter tous les items dans la séquence d'entrée le nombre de fois spécifié dans l'argument **count**. Par exemple, si le décompte est **2**, la séquence `1,2,3` produit `1,2,3,1,2,3`.



Langages

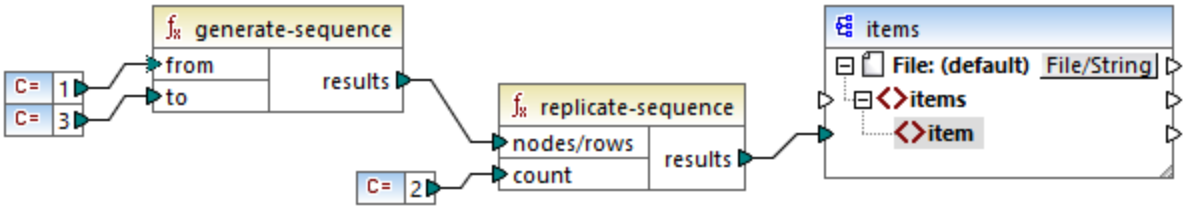
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Description
node/rows	Cette entrée doit recevoir une connexion depuis un item de mappage qui fournit une séquence[406] de zéros ou plus de valeurs. Par exemple, la connexion peut provenir d'un item XML de source .
count	Spécifie le nombre de fois nécessaire pour répliquer la séquence connectée.

Exemple

Le mappage fictif suivant génère la séquence `1,2,3`. La séquence est traitée par la fonction `replicate-sequence` et le résultat est écrit dans un fichier XML de cible.

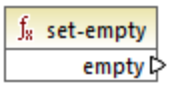


Puisque l'argument **count** est défini sur **2**, la séquence est répliquée deux fois puis est transmise sur la cible. Par conséquent, la sortie de mappage est la suivante (à l'exception des déclarations XML et de schéma) :

```
<items>
   <item>1</item>
   <item>2</item>
   <item>3</item>
   <item>1</item>
   <item>2</item>
   <item>3</item>
</items>
```

6.6.9.17 set-empty

Retourne une séquence vide.

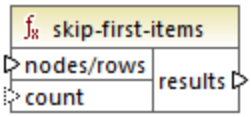


Langages

Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

6.6.9.18 skip-first-items

Saute les premiers items/nœuds "N" de la séquence d'entrée, où N est le nombre fourni par le paramètre **count** et retourne le reste de la séquence.



Langages

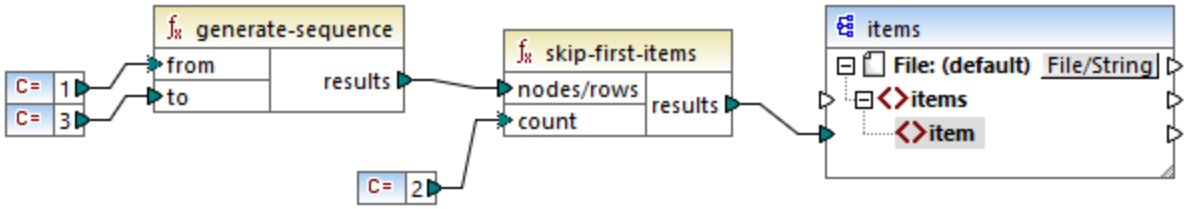
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Description
node/rows	Cette entrée doit recevoir une connexion depuis un item de mappage qui fournit une séquence[406] de zéros ou plus de valeurs. Par exemple, la connexion peut provenir d'un item XML de source .
count	Argument optionnel. Spécifie le nombre d'items à sauter. La valeur par défaut est de **1**.

Exemple

Le mappage fictif suivant génère la séquence `1,2,3`. La séquence est traitée par la fonction `skip-first-items` et le résultat est écrit dans un fichier XML de cible.



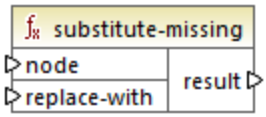
Puisque l'argument **count** est défini sur **2**, les premiers deux items sont sautés et les items restants sont transférés à la cible. Par conséquent, la sortie de mappage est la suivante (à l'exception des déclarations XML et de schéma) :

```
<items>
   <item>3</item>
```

```
</items>
```

6.6.9.19 substitute-missing

Cette fonction est une combinaison pratique de la fonction de exists[278] et if-else condition[179]. Si l'item connecté à l'entrée de **node** existe, son contenu sera copié dans la cible. Sinon, le contenu de l'item connecté à l'entrée de **replace-with** sera copié dans la cible.



Langages

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Paramètres

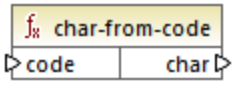
Nom	Description
node	Cette entrée doit recevoir une connexion depuis un item de mappage qui fournit une séquence[406] de zéros ou plus de valeurs. Par exemple, la connexion peut provenir d'un item XML de source .
replace-with	Cette entrée doit recevoir une connexion depuis un item de mappage qui fournit la valeur de remplacement

6.6.10 core | string functions

Les fonctions string vous permettent de manipuler des données de string pour extraire des portions de string, tester des sous-string, extraire l'information concernant des strings, partager des strings, etc.

6.6.10.1 char-from-code

Retourne la représentation de caractère de la valeur décimale Unicode (code) fournie en tant qu'argument.
Astuce : Pour trouver le code décimal Unicode d'un caractère, vous pouvez utiliser la fonction **code-from-char**[306].



Langages

Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

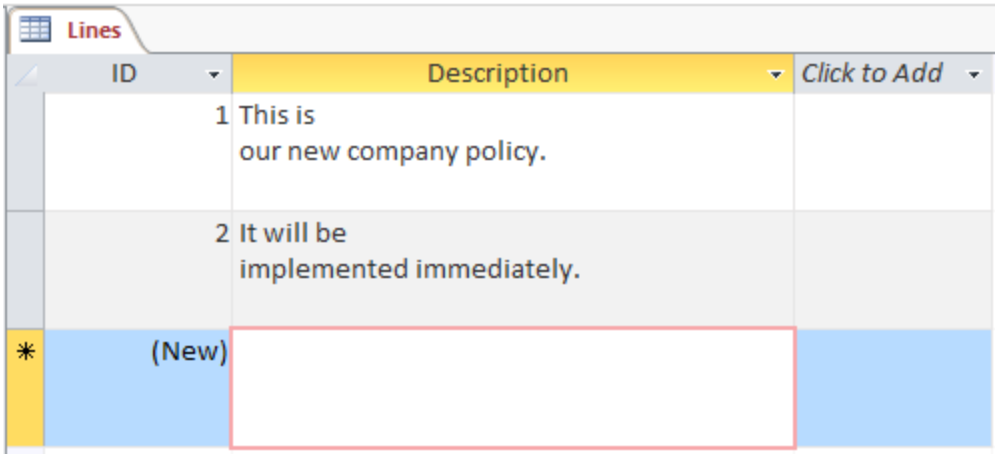
Nom	Description
code	La valeur Unicode, en tant que nombre décimal.

Exemple 1

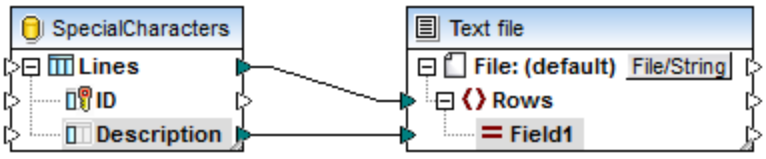
Conformément aux graphiques disponibles sur le site web Unicode (https://www.unicode.org/charts/), le point d'exclamation a la valeur hexadécimale de `0021`. La valeur correspondante en format décimal est **33**. C'est pourquoi, si vous fournissez `33` en tant qu'argument à la fonction `char-from-code`, le caractère `!` Sera retourné.

Exemple 2 (éditions Professional et Enterprise)

Cet exemple affiche comment remplacer les caractères spéciaux dans une base de données avec des espaces. Considérez une base de données SQLite consistant en une table "Lignes" qui a deux colonnes : "ID" et "Description".



L'objectif est d'extraire chaque description d'un fichier CSV (une description par ligne) ; ainsi un mappage pour atteindre cet objectif pourrait ressembler à l'exemple suivant :



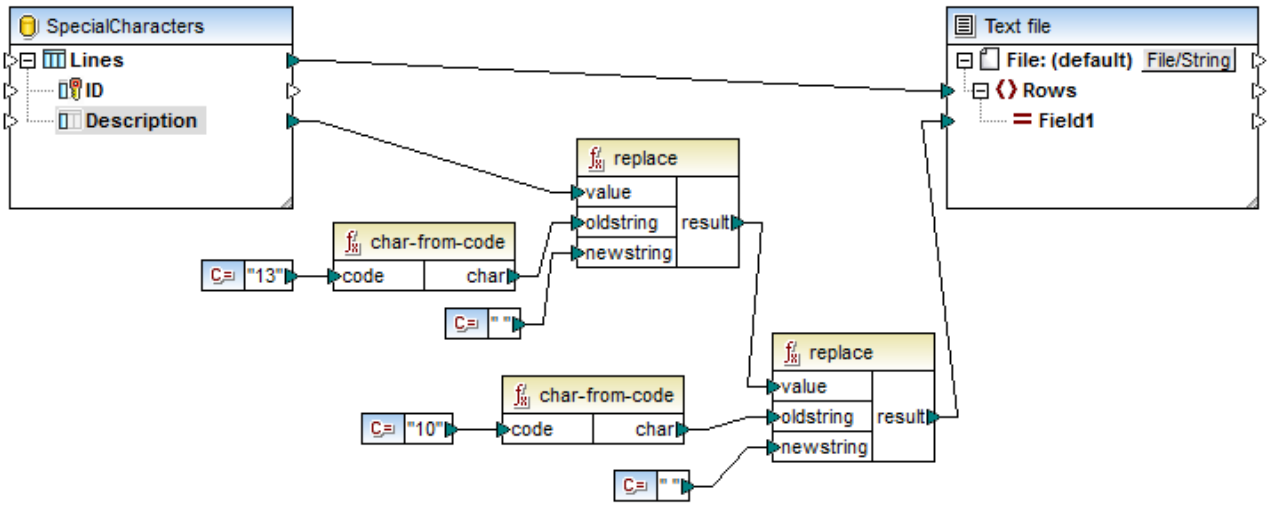
Néanmoins, étant donné que chaque ligne "Description" dans Access contient plusieurs lignes séparées par des caractères CR/LF, la sortie de mappage contient aussi des sauts de ligne, ce qui n'est pas le résultat intenté :

```
"This is
our new company policy."
"It will be
implemented immediately."

```

Afin de surmonter ce problème, nous allons ajouter au mappage les fonctions **char-from-code** et **replace** depuis la bibliothèque intégrée de MapForce. Chaque description doit être traitée de manière à ce que, quels que soient les caractères rencontrés ci-dessus, ils doivent être remplacés par un caractère d'espace.

Dans le graphique Unicode (http://www.unicode.org/charts/), les caractères LF et CR correspondent aux caractères **hex 0A | dec 10** et **hex 0D | dec 13**, respectivement. C'est pourquoi le mappage doit être modifié pour convertir les valeurs Unicode décimales 13 et 10 en un string, afin de permettre un autre traitement par la fonction **replace**.



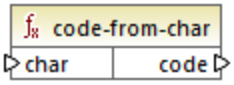
Si vous faites un aperçu du mappage maintenant, vous verrez que les caractères CR/LF se trouvant dans le champ de base de données ont chacun été remplacés par un espace.

```
This is  our new company policy.
It will be  implemented immediately.

```

6.6.10.2 code-from-char

Retourne la valeur décimale Unicode (code) du caractère fourni en tant qu'argument. Si le string fourni en tant qu'argument a plusieurs caractères, le code du premier caractère est retourné.



Langages

Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

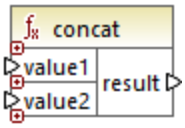
Nom	Description
char	La valeur de string d'entrée.

Exemple

Si l'entrée**char** est le caractère **$** (signe de dollar), la fonction retourne **36** (qui est la valeur Unicode décimale pour ce caractère).

6.6.10.3 concat

Concatène (ajoute) deux ou plusieurs valeurs dans un seul string de résultat. Toutes les valeurs d'entrée sont converties automatiquement dans le type "string". Par défaut, cette fonction a uniquement deux paramètres, mais vous pouvez en ajouter plus. Cliquer sur **Ajouter paramètres** (⊞) ou **Supprimer paramètre** (⊠) pour ajouter ou supprimer des paramètres.



Note: Toutes les entrées de la fonction `concat` doivent avoir une valeur. Si une des entrées n'a pas de valeur, la fonction ne sera pas appelée et une erreur se produira. Veuillez noter qu'un string vide est une valeur d'entrée valide ; néanmoins, une séquence vide (comme le résultat de la fonction **`set-empty`**) n'est pas une valeur valide et en conséquence la fonction échouera. Pour éviter cette situation, vous pouvez d'abord traiter les valeurs avec la fonction substitute-missing 304 puis fournir le résultat en tant qu'entrée dans la fonction **`concat`**.

Langages

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

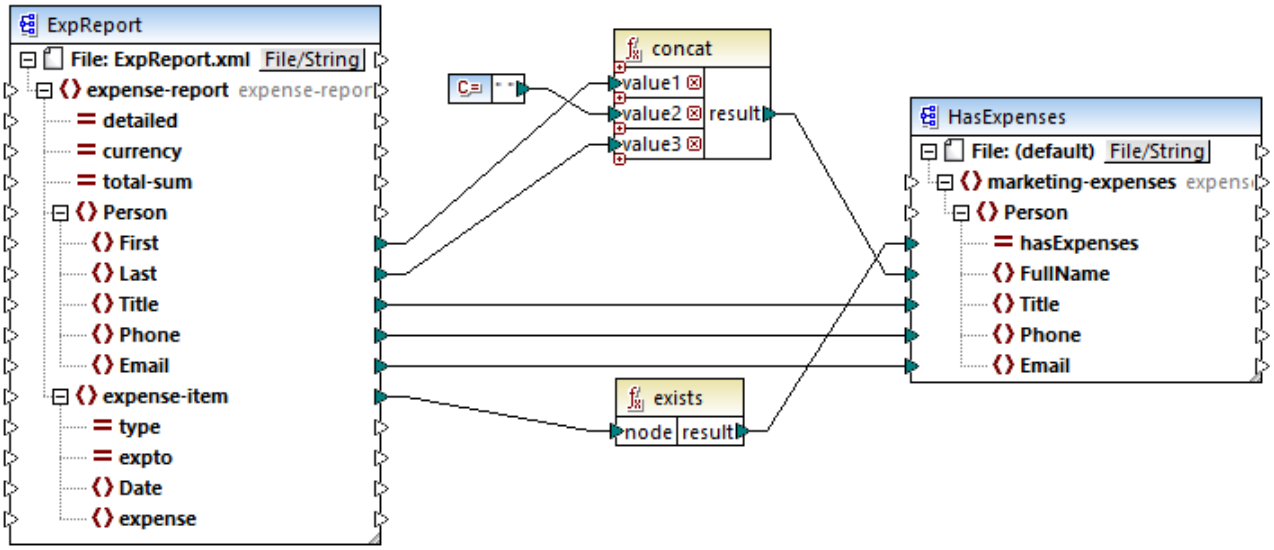
Paramètres

Nom	Description
value1	La première valeur d'entrée.

Nom	Description
value2	La deuxième valeur d'entrée.
valueN	La *N* valeur d'entrée.

Exemple

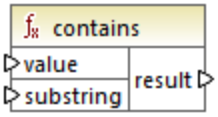
Dans le mappage illustré ci-dessous, la fonction **concat** réunit le prénom, la constante " " et le nom de famille. La valeur de retour est ensuite écrite dans l'item de cible **FullName**. Le mappage de cette fonction est disponible dans le chemin suivant :
<Documents>\Altova\MapForce2024\MapForceExamples\HasMarketingExpenses.mfd.



HasMarketingExpenses.mfd

6.6.10.4 contains

Retourne Booléenne **true** si la valeur de string fournie en tant qu'argument contient le sous-string fourni en tant qu'argument.



Langages

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Paramètres

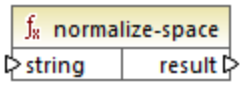
Nom	Description
value	La valeur d'entrée (c'est à dire la "pile de foin").
substring	Le sous-string à chercher (c'est à dire l'"aiguille").

Exemple

Si la **value** d'entrée est "category" et **substr** est "cat", la fonction retourne **true**.

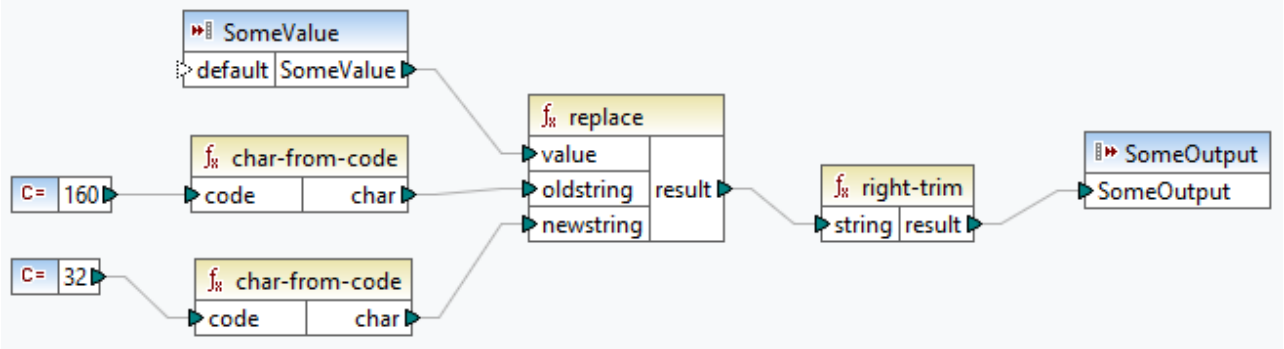
6.6.10.5 normalize-space

La fonction **`normalize-space`** (*voir la capture d'écran ci-dessous*) supprime des espaces à gauche et à droite d'un string et remplace les espaces blancs internes avec un espace blanc unique. Les espaces blancs incluent un espace (U+0020), un onglet (U+0009), le retour à la ligne (U+000D) et la saut de ligne (U+000A). Pour des détails sur les espaces blancs, voir la Recommandation XML.



À propos des espaces insécables

Les fonctions **`left-trim`**, **`right-trim`** et **`normalize-space`** ne suppriment pas les espaces insécables. L'une des solutions possibles pourrait être de remplacer l'espace insécable, dont la représentation est 160, avec l'espace, dont la représentation décimale est 32. Le mappage ci-dessous montre qu'une fois que l'espace insécable a été remplacé, la valeur découpée `SomeValue` sera mappée vers la cible.



Si votre composant source est un fichier Excel, vous pouvez supprimer les espaces supplémentaires dans Excel utilisant une combinaison de fonctions TRIM, CLEAN et SUBSTITUTE. Pour les détails, voir *Supprimer les espaces et caractères non imprimables du texte*.

Langages

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Paramètres

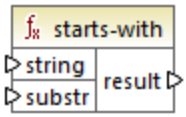
Nom	Description
string	Le string d'entrée à normaliser.

Exemple

Si le string d'entrée est `The quick  brown fox`, la fonction retourne `The quick brown fox`.

6.6.10.6 starts-with

Retourne Booléenne **true** si le string fourni en tant qu'argument commence avec le sous-string fourni en tant qu'argument; **false** sinon.



Langages

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Paramètres

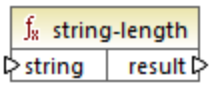
Nom	Description
string	Type
substr	Le sous-string à vérifier.

Exemple

Si le **string** d'entrée est `category` et **substr** est `cat`, la fonction retourne **true**.

6.6.10.7 string-length

Retourne le nombre de caractères dans le string fourni en tant qu'argument.



Langages

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Paramètres

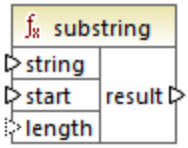
Nom	Description
string	Type

Exemple

Si le string d'entrée est `car`, la fonction retourne **3**. Si le string d'entrée est string vide, la fonction retourne **0**.

6.6.10.8 substring

Retourne la portion du string spécifié par les paramètres **start** et **length**.



Langages

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Paramètres

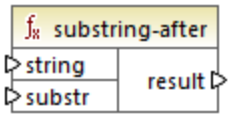
Nom	Description
string	Type
start	Spécifie la position de démarrage (index) à partir de laquelle le sous-string.devrait être extrait. Le premier index est **1**.
length	Optionnel Spécifie le nombre de caractères à extraire. Si le paramètre **length** n'est pas spécifié, le résultat est un fragment commençant par **start** jusqu'à la fin du string.

Exemple

Si le string d'entrée est `MapForce`, le commencement est **1** et la longueur est **3**, la fonction retourne `Map`. Si le string d'entrée est `MapForce`, le commencement est **4** et la longueur n'est pas fournie, la fonction retourne `Force` .

6.6.10.9 substring-after

Retourne la portion du string qui se produit après la première occurrence de **substr**. Si **substr** ne se produit pas dans **string**, la fonction retourne un string vide.



Langages

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Paramètres

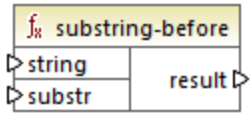
Nom	Description
string	Le string d'entrée
substr	Le sous-string. Tout caractère après la première occurrence de **substr** est le résultat de la fonction.

Exemple

Si le string d'entrée est `MapForce` et **substr** est `Map`, la fonction retourne `Force`. Si le string d'entrée est `2020/01/04` et **substr** est `/`, la fonction retourne `01/04`.

6.6.10.10 substring-before

Retourne la portion du string qui se produit avant la première occurrence de **substr**. Si **substr** ne se produit pas dans **string**, la fonction retourne un string vide.



Langages

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Description
string	Type

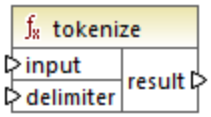
Nom	Description
substr	Le sous-string. Tout caractère avant la première occurrence de **substr** est le résultat de la fonction.

Exemple

Si le string d'entrée est `MapForce` et **substr** est `Force`, la fonction retourne `Map`. Si le string d'entrée est `2020/01/04` et **substr** est `/`, la fonction retourne `2020`.

6.6.10.11 tokenize

Partage le string d'entrée en une séquence de strings en utilisant le délimiteur fourni en tant qu'argument.



Langages

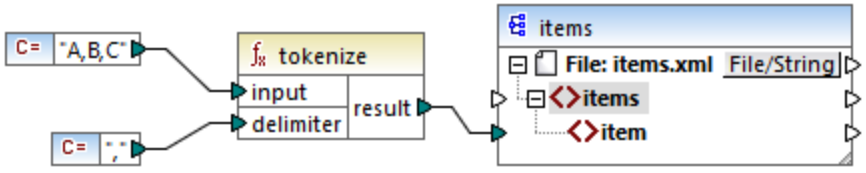
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Description
input	Type
delimiter	Le délimiteur à utiliser.

Exemple

Si le string d'entrée est `A,B,C` et le délimiteur est `,` alors la fonction retourne une séquence de trois strings : `A`, `B` et `C`.

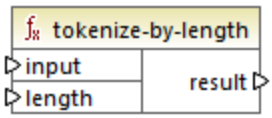


Dans le mappage fictif illustré ci-dessus, le résultat de la fonction est une séquence de strings. Conformément au mappage général rules 406, pour chaque item dans la séquence de source, un nouvel **item** est créé dans le composant de cible. Par conséquent, la sortie de mappage ressemble à :

```
<items>
   <item>A</item>
   <item>B</item>
   <item>C</item>
</items>
```

6.6.10.12 tokenize-by-length

Partage le string d'entrée dans une séquence de strings. La taille de chaque string résultant est déterminé par le paramètre **length**.



Langages

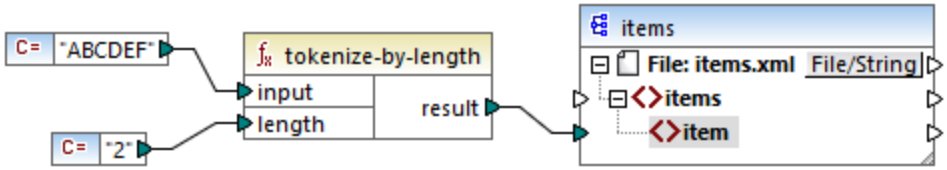
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Description
input	Type
length	Détermine la longueur de chaque string dans la séquence générée des strings.

Exemple

Si le string d'entrée est **ABCDEF** et que la longueur est **2**, alors la fonction retourne une séquence de trois strings : **AB**, **CD**, et **EF**.



Dans le mappage fictif illustré ci-dessus, le résultat de la fonction est une séquence de strings. Conformément au mappage général rules 406, pour chaque item dans la séquence de source, un nouvel **item** est créé dans le composant de cible. Par conséquent, la sortie de mappage ressemble à :

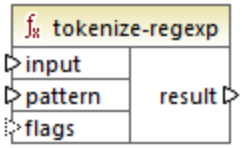
```
<items>
   <item>AB</item>
   <item>CD</item>
```

```
   <item>EF</item>
</items>
```

6.6.10.13 tokenize-regexp

Partage le string d'entrée dans une séquence de strings. Tout sous-string qui correspond au **pattern** d'expression régulier fourni en tant qu'argument définit le séparateur. Les strings correspondants (séparateur) ne sont pas inclus dans le résultat retourné par la fonction.

Note : Lors de la génération de code C++, C# ou Java, les fonctions avancées de la syntaxe d'expression régulière peuvent différer légèrement. Voir la documentation regex de chaque langage pour plus d'informations.



Langages

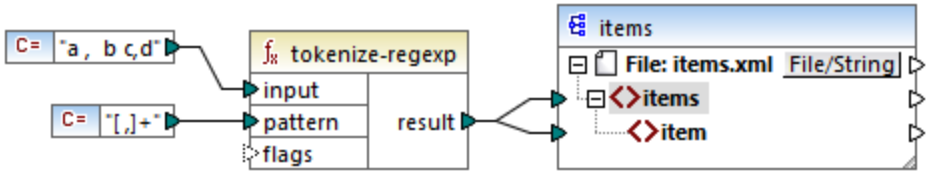
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Description
input	Type
pattern	Apporte un pattern d'expression régulier. Tout sous-string qui correspond au pattern sera traité en tant que délimiteur. Pour plus d'informations, voir Expressions régulières 228.
flags	Paramètre optionnel. Fournit l'expression régulière flags 230 à utiliser. Par exemple, le flag "i" instruit le processus de mappage de fonctionner dans le mode insensible à la casse.

Exemple

L'objectif du mappage illustré ci-dessous est de partager le string `a ,  b c,d` dans une séquence des strings, où chaque caractère alphabétique est un item dans la séquence. Tout espace blanc ou virgule redondant doit être supprimé.



Pour obtenir cet objectif, le pattern d'expression régulière `[ ,]+` a été fourni en tant que paramètre dans la fonction `tokenize-regexp`. Ce pattern a la signification suivante :

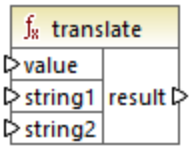
- Il correspond à un des caractères dans la classe de caractère `[ ,]`. Ainsi, un partage se produira à chaque fois qu'une virgule ou qu'une espace est rencontrée dans le string d'entrée.
- Le quantificateur `+` spécifie qu'une ou plusieurs occurrences de la classe du caractère précédent doivent trouver correspondance. Sans ce quantificateur, chaque occurrence d'espace ou de virgule créera un item séparé dans la séquence résultante de strings, ce qui n'est pas le résultats intenté.

Le sortie de mappage est la suivante :

```
<items>
   <item>a</item>
   <item>b</item>
   <item>c</item>
   <item>d</item>
</items>
```

6.6.10.14 translate

Effectue un remplacement caractère par caractère. Il regarde dans la **value** pour les caractères contenus dans **string1**, et remplace chaque caractère avec celui dans la même position dans le **string2**. Lorsqu'il n'y a pas de caractères correspondants dans **string2**, le caractère est supprimé.



Langages

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

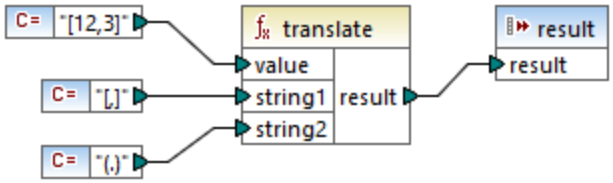
Paramètres

Nom	Description
value	Type

Nom	Description
string1	Fournit une liste des caractères de recherche. La position de chaque caractère se trouvant dans le string est importante.
string2	Fournit une liste de caractères de remplacement. La position de chaque caractère de remplacement doit correspondre à celui dans **string1**.

Exemple

Supposons que vous souhaitez convertir de string `[12,3]` en `(12.3)`. Concrètement, les crochets doivent être remplacés par des parenthèses, et toute virgule doit être remplacée par le caractère de point. Pour ce faire vous pouvez appeler la fonction `translate` comme suit :



Dans le mappage ci-dessus, la première constante fournit le string d'ouverture à traiter. La seconde et la troisième constante fournit une liste des caractères en tant que **string1** et **string2**, respectivement.

string1 `[,]`

string2 `(.)`

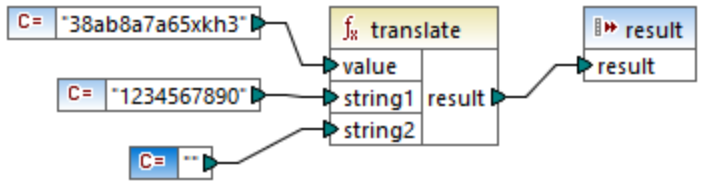
Veuillez noter que les deux **string1** et **string2** ont le même nombre de caractères. Pour chaque caractère dans **string1**, le caractère équivalent à la même position depuis **string2** sera utilisé en tant que remplacement. Par conséquent, le remplacement suivant aura lieu :

- Chaque `[` sera remplacé par un `(`
- Chaque `,` sera remplacé par un `.`
- Chaque `]` sera remplacé par un `]`

Le sortie de mappage est la suivante :

```
(12.3)
```

Cette fonction peut aussi être utilisée pour retirer certains caractères sélectivement depuis un string. Pour ce faire, définir le paramètre **string1** sur les caractères que vous souhaitez supprimer, et **string2** sur un string vide. Par exemple, le mappage ci-dessous supprime tous les chiffres depuis le string `38ab8a7a65xkh3`.



Le sortie de mappage est la suivante :

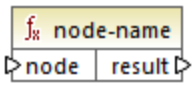
```
abaaxkh
```

6.6.11 xpath2 | accessors

Les fonctions provenant de la sous-bibliothèque **xpath2 | accessors** extraient des informations concernant des nœuds ou des items XML. Ces fonctions sont disponibles lorsque les langages XSLT2 ou XQuery sont sélectionnés.

6.6.11.1 base-uri

La fonction **base-uri** prend un nœud en tant qu'entrée et retourne l'URI de la ressource XML contenant le nœud. La sortie est de type `xs:string`.



Langages

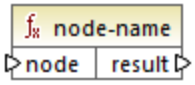
XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Type	Description
node	`mf:node`	Le nœud d'entrée.

6.6.11.2 node-name

La fonction **node-name** prend un nœud en tant que son argument d'entrée et retourne son QName. Lorsque le QName est représenté en tant que string, il prend la forme de `prefix:localname` si le nœud prend un préfixe, ou `localname` si le nœud n'a pas de préfixe. Pour obtenir l'URI d'espace de noms d'un nœud, utiliser la fonction **namespace-uri-from-QName** 275.



Langages

XQuery, XSLT 2.0, XSLT 3.0.

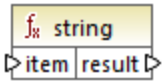
Paramètres

Nom	Type	Description
node	`mf:node`	Le nœud d'entrée.

6.6.11.3 string

La fonction **`string`** fonctionne comme le constructeur `xs:string` : elle convertit son argument en `xs:string`.

Lorsque l'argument d'entrée est une valeur d'un type atomique (par exemple `xs:decimal`), cette valeur atomique est convertie en une valeur de type `xs:string`. Si l'argument d'entrée est un nœud, la valeur string du nœud est extraite. (La valeur de string d'un nœud est une concaténation des valeurs des nœuds descendants du nœud.)



Langages

XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Type	Description
item	`mf:item`	La valeur d'entrée.

6.6.12 xpath2 | anyURI functions

La sous-librairie **xpath2 | anyURI** contient la fonction **`resolve-uri`**. Cette fonction est disponible lorsque les langages XSLT2 ou XQuery sont sélectionnés.

6.6.12.1 resolve-uri

La fonction **resolve-uri** prend un URI en tant que son premier argument et le résout par rapport à l'URI dans le second argument. Le résultat est de type de données `xs:string`. La mise en place de la fonction traite les deux entrées en tant que strings ; aucun contrôle n'est effectué pour voir si les ressources identifiées par ces URIs existent réellement.

Langages

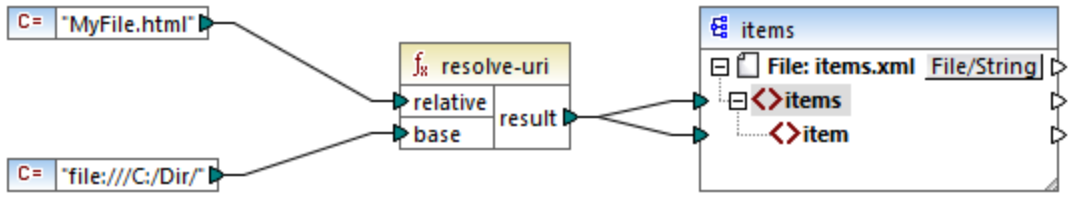
XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Type	Description
relative	**xs:string**	L'URI relatif à résoudre par rapport à la base.
base	**xs:string**	L'URI de base.

Exemple

Dans le mappage illustré ci-dessous, le premier argument fournit l'URI relative **MyFile.html**, et le second argument fournit l'URI de base **file:///C:/Dir/**. L'URI résolu sera une concaténation des deux, donc **file:///C:/Dir/MyFile.html**.

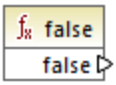


6.6.13 xpath2 | boolean functions

Les fonctions booléennes **true** et**false** ne prennent pas d'argument et retournent les valeurs de constante booléennes, **true** et **false**, respectivement. Elles peuvent être utilisées là où une valeur booléennen constante est requise.

6.6.13.1 false

Retourne la valeur booléenne **false**.

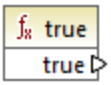


Langages

XQuery, XSLT 2.0, XSLT 3.0.

6.6.13.2 true

Retourne la valeur booléenne **true**.



Langages

XQuery, XSLT 2.0, XSLT 3.0.

6.6.14 xpath2 | constructors

Les fonctions dans la partie "constructors" de la sous-bibliothèque de la bibliothèque XPath 2.0 construisent des types de données spécifiques depuis le texte d'entrée. La table suivante liste les fonctions de constructor disponibles.

xs:ENTITY	**xs:double**	**xs:nonPositiveInteger**
xs:ID	**xs:duration**	**xs:normalizedString**
xs:IDREF	**xs:float**	**xs:positiveInteger**
xs:NCName	**xs:gDay**	**xs:short**
xs:NMTOKEN	**xs:gMonth**	**xs:string**
xs:Name	**xs:gMonthDay**	**xs:time**
xs:QName	**xs:gYear**	**xs:token**
xs:anyURI	**xs:gYearMonth**	**xs:unsignedByte**
xs:base64Binary	**xs:hexBinary**	**xs:unsignedInt**

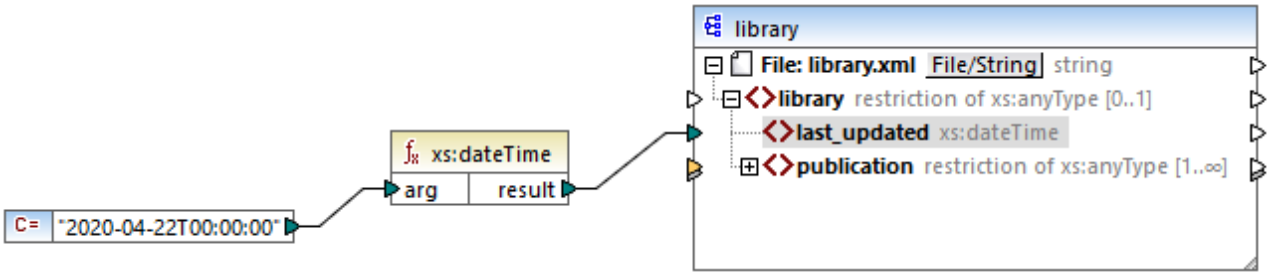
xs:boolean	**xs:int**	**xs:unsignedLong**
xs:byte	**xs:integer**	**xs:unsignedShort**
xs:date	**xs:language**	**xs:untypedAtomic**
xs:dateTime	**xs:long**	**xs:yearMonthDuration**
xs:dayTimeDuration	**xs:negativeInteger**	
xs:decimal	**xs:nonNegativeInteger**	

Langages

XQuery, XSLT 2.0, XSLT 3.0.

Exemple

Généralement, le format lexical du texte d'entrée doit être celui prévu par le type de données à construire. Sinon, la transformation ne sera pas réussie Par exemple, si vous souhaitez construire une valeur `xs:dateTime` en utilisant la fonction de constructor **`xs:dateTime`**, le texte d'entrée doit avoir le format lexical du type de données `xs:dateTime`, qui est **`YYYY-MM-DDTHH:mm:ss`**.



Dans le mappage illustré ci-dessus, une constante de string (**`"2020-04-28T00:00:00"`**) a été utilisée pour fournit l'argument d'entrée de la fonction. L'entrée peut aussi avoir été obtenue depuis un item dans le document source. La fonction **`xs:dateTime`** retourne la valeur **`2020-04-28T00:00:00`** de type `xs:dateTime`.

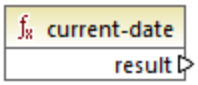
Pour consulter le type de données attendu d'un item de mappage (y compris le type de données des arguments de fonction), déplacer le curseur de la souris sur le connecteur d'entrée ou de sortie respectif.

6.6.15 xpath2 | context functions

Les fonctions context provenant de la bibliothèque **xpath2** fournissent des information diverses à propos de la date et de l'heure actuels, la collation par défaut utilisée par le processeur, la taille de la séquence actuelle et la position du nœud actuel.

6.6.15.1 current-date

Retourne la date actuelle (`xs:date`) depuis l'horloge du système.

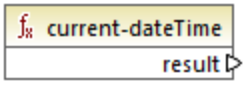


Langages

XQuery, XSLT 2.0, XSLT 3.0.

6.6.15.2 current-dateTime

Retourne la date et l'heure actuels (`xs:dateTime`) depuis l'horloge du système.

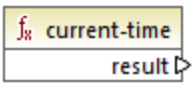


Langages

XQuery, XSLT 2.0, XSLT 3.0.

6.6.15.3 current-time

Retourne l'heure actuelle (`xs:time`) depuis l'horloge du système.



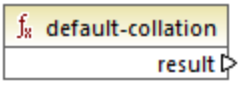
Langages

XQuery, XSLT 2.0, XSLT 3.0.

6.6.15.4 default-collation

La fonction **`default-collation`** ne prend pas d'argument et retourne la collation par défaut, c'est à dire la collation qui est utilisée lorsqu'aucune collation n'est spécifiée pour une fonction où une peut être spécifiée.

Des comparaisons, y compris pour les fonctions **`max-string`** et **`min-string`** sont basées sur cette collation.

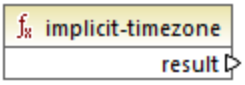


Langages

XQuery, XSLT 2.0, XSLT 3.0.

6.6.15.5 implicit-timezone

Retourne la valeur de la propriété "implicit timezone" depuis le contexte d'évaluation.

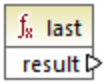


Langages

XQuery, XSLT 2.0, XSLT 3.0.

6.6.15.6 last

Retourne la nombre d'items dans le cadre de la séquence des items actuellement en cours de traitement. Chose importante, la séquence des items est déterminée par le contexte de mappage[407] actuel, tel que décrit dans l'exemple ci-dessous.



Langages

XQuery, XSLT 2.0, XSLT 3.0.

Exemple

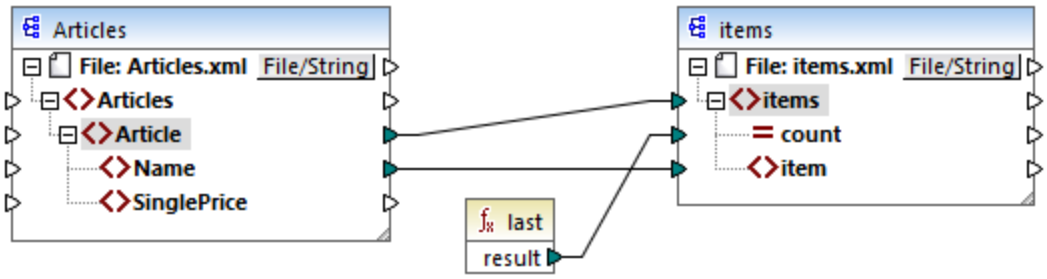
Supposons que vous disposez du fichier XML de source suivant :

```
<Articles>
   <Article>
      <Name>T-Shirt</Name>
      <SinglePrice>25</SinglePrice>
   </Article>
   <Article>
      <Name>Socks</Name>
```



```
      <SinglePrice>2.30</SinglePrice>
   </Article>
   <Article>
      <Name>Jacket</Name>
      <SinglePrice>57.50</SinglePrice>
   </Article>
</Articles>
```

Votre objectif est de copier des données dans un fichier XML avec un schéma différent. De plus, le décompte de tous les items doit être enregistré dans le fichier XML de cible. Cela peut être obtenu par un mappage comme celui ci-dessous :



Dans l'exemple ci-dessus, la fonction **last** retourne la position du dernier nœud dans le contexte de parent actuel et remplit l'attribut **count** avec la valeur **3**.

```
<items count="3">
   <item>T-Shirt</item>
   <item>Pants</item>
   <item>Jacket</item>
</items>
```

Veuillez noter que la valeur **3** est la position du dernier item (et donc le décompte de tous les items) dans le contexte de mappage créé par la connexion entre **Article** et **items**. Si cette connexion n'existait pas, les items seraient toujours copiés vers la cible, mais la fonction **last** retournerait la valeur **1** de manière incorrecte, parce qu'il n'y aurait pas de parent context 412 à itérer. (Plus précisément, elle utiliserait le contexte implicité par défaut créé entre les items racine des deux composants, ce qui produit une séquence de 1 item, et pas de 3 comme attendu).

Il est généralement conseillé d'utiliser la fonction **count** 236 depuis la bibliothèque **core** au lieu de la fonction **last**, parce que cette première a un argument **parent-context**, ce qui vous permet de modifier le contexte de mappage explicitement.

6.6.16 xpath2 | durations, date and time functions

Les fonctions date and time de la bibliothèque **xpath2** vous permettent d'ajuster le fuseau horaire dans valeurs de dates et d'heures, d'extraire des composants particuliers depuis les valeurs date, time et duration, et de soustraire des valeurs date and time.

Ajuster le fuseau horaire

Pour ajuster le fuseau horaire dans des valeurs date and time, les fonctions suivantes sont disponibles :

- **adjust-date-to-timezone**
- **adjust-date-to-timezone** (avec argument timezone)
- **adjust-dateTime-to-timezone**
- **adjust-dateTime-to-timezone** (avec argument timezone)
- **adjust-time-to-timezone**
- **adjust-time-to-timezone** (avec argument timezone)

Chacune de ces fonctions liées prennent une valeur `xs:date`, `xs:time` ou `xs:dateTime` en tant que le premier argument et ajuste l'entrée en ajoutant, supprimant ou en modifiant le composant de fuseau horaire selon la valeur du second argument (le cas échéant).

Les situations suivantes sont possibles lorsque le premier argument ne continent aucun fuseau horaire (par exemple, la date `2020-01` ou l'heure `14:00:00`).

- Si l'argument **timezone** est présent, le résultat contiendra le fuseau horaire spécifié dans le second argument. Le fuseau horaire dans le second argument est ajouté.
- Si l'argument **timezone** est absent, le résultat contiendra le fuseau horaire implicite qui est le fuseau horaire du système. Le fuseau horaire du système est ajouté.
- Si l'argument **timezone** est vide, le résultat ne contiendra aucun fuseau horaire.

Les situations suivantes sont possibles lorsque le premier argument contient aucun fuseau horaire (par exemple, la date `2020-01-01+01:00` ou l'heure `14:00:00+01:00`).

- Si l'argument **timezone** est présent, le résultat contiendra le fuseau horaire spécifié dans le second argument. Le fuseau horaire original est remplacé par le fuseau horaire dans le second argument.
- Si l'argument **timezone** est absent, le résultat contiendra le fuseau horaire implicite qui est le fuseau horaire du système. Le fuseau horaire original est remplacé par le fuseau horaire du système.
- Si l'argument **timezone** est vide, le résultat ne contiendra aucun fuseau horaire.

Extraire des composants de dates et d'heures

Pour extraire des valeurs numériques comme des heures, des minutes, des jours, des mois, etc. depuis des valeurs de date et d'heure, les fonctions suivantes sont disponibles :

- **day-from-date**
- **day-from-dateTime**
- **hours-from-dateTime**
- **hours-from-time**
- **minutes-from-dateTime**
- **minutes-from-time**
- **month-from-date**
- **month-from-dateTime**
- **seconds-from-dateTime**
- **seconds-from-time**
- **timezone-from-date**
- **timezone-from-dateTime**
- **timezone-from-time**

- **year-from-date**
- **year-from-dateTime**

Chacune de ces fonctions extrait un composant particulier des valeurs `xs:date`, `xs:time`, `xs:dateTime` et `xs:duration`. Le résultat sera soit `xs:integer` soit `xs:decimal`.

Extraire des composants de durations

Pour extraire des composants time depuis des durées, les fonctions suivantes sont disponibles :

- **days-from-duration**
- **hours-from-duration**
- **minutes-from-duration**
- **months-from-duration**
- **seconds-from-duration**
- **years-from-duration**

La durée doit être spécifiée soit en tant que `xs:yearMonthDuration` (pour extraire les années et les mois) ou `xs:dayTimeDuration` (fpour extraire des jours, des heures, des minutes et des secondes). Toutes les fonctions retournent un résultat de type `xs:integer`, avec l'exception de la fonction **seconds-from-duration**, qui retourne `xs:decimal`.

Soustraire des valeurs de date et d'heure

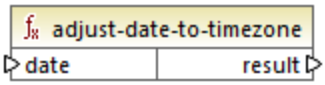
Pour soustraire des valeurs de date et d'heure, les fonctions suivantes sont disponibles :

- **subtract-dateTimes**
- **subtract-dates**
- **subtract-times**

Chacune des fonctions de soustraction vous permet de soustraire une valeur d'heure d'une autre et de retourner une valeur de durée

6.6.16.1 adjust-date-to-timezone

Ajuste une valeur `xs:date` dans le fuseau horaire implicite dans le contexte d'évaluation (le fuseau horaire du système).



Langages

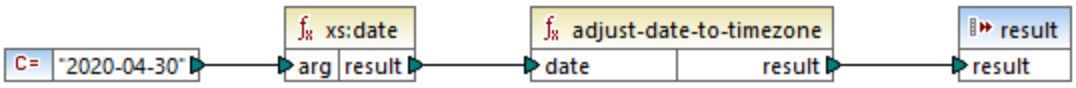
XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Type	Description
date	**xs:date**	La valeur d'entrée de type xs:date.

Exemple

Le mappage suivant construit un xs:date depuis un string et le fournit en tant qu'argument à la fonction **adjust-date-to-timezone**.

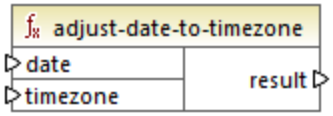


Mappage XSLT 2.0

Si le mappage est exécuté sur un ordinateur situé dans un fuseau horaire de +02:00, la fonction adjuste la valeur de date pour inclure le fuseau horaire du système. Par conséquent, la sortie de mappage est **2020-04-30+02:00**.

6.6.16.2 adjust-date-to-timezone

Ajuste une valeur xs:date dans un fuseau horaire spécifique ou aucun fuseau horaire. Si l'argument **timezone** est une séquence vide, la fonction retourne un xs:date sans fuseau horaire. Sinon, il retourne un xs:date avec un fuseau horaire.



Langages

XQuery, XSLT 2.0, XSLT 3.0.

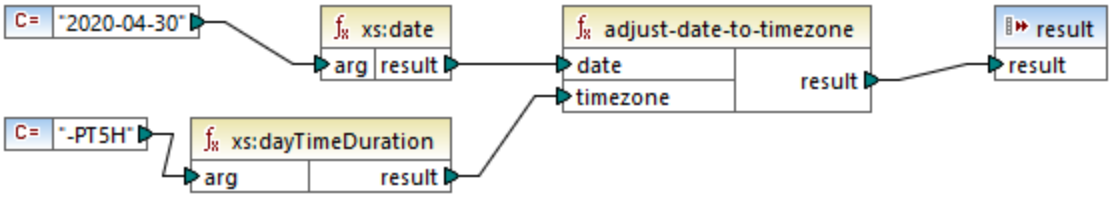
Paramètres

Nom	Type	Description
date	**xs:date**	La valeur d'entrée de type xs:date.
timezone	**xs:dayTimeDuration**	Le fuseau horaire exprimé en tant que valeur xs:dayTimeDuration. La valeur peut être négative. Par

Nom	Type	Description
		exemple, une valeur de fuseau horaire de -5 heures peut être exprimée en tant que `-PT5H`.

Exemple

Le mappage suivant construit les deux paramètres dans la fonction **`adjust-date-to-timezone`** depuis des strings, en utilisant les fonction XPath 2 constructor[321] correspondantes. L'objectif du mappage est d'ajuster le fuseau horaire à -5 heures. Ce fuseau horaire peut être exprimé en tant que `-PT5H`.

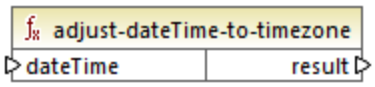


Mappage XSLT 2.0

La fonction ajuste la valeur de date au fuseau horaire fourni en tant qu'argument. Par conséquent, la sortie de mappage est `2020-04-30-05:00`.

6.6.16.3 adjust-dateTime-to-timezone

Ajuste une valeur `xs:dateTime` dans le fuseau horaire implicite dans le contexte d'évaluation (le fuseau horaire du système).



Langages

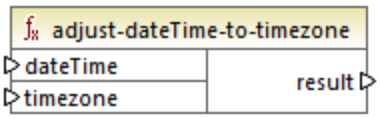
XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Type	Description
dateTime	`xs:dateTime`	La valeur d'entrée de type `xs:dateTime`.

6.6.16.4 adjust-dateTime-to-timezone

Ajuste une valeur `xs:dateTime` dans un fuseau horaire spécifique ou aucun fuseau horaire. Si l'argument **timezone** est une séquence vide, la fonction retourne un `xs:dateTime` sans fuseau horaire. Sinon, il retourne un `xs:dateTime` avec un fuseau horaire.



Langages

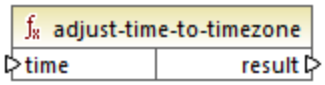
XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Type	Description
dateTime	**xs:dateTime**	La valeur d'entrée de type `xs:dateTime`.
timezone	**xs:dayTimeDuration**	Le fuseau horaire exprimé en tant que valeur `xs:dayTimeDuration`. La valeur peut être négative. Par exemple, une valeur de fuseau horaire de -5 heures peut être exprimée en tant que **-PT5H**.

6.6.16.5 adjust-time-to-timezone

Ajuste une valeur `xs:time` dans le fuseau horaire implicite dans le contexte d'évaluation (le fuseau horaire du système).



Langages

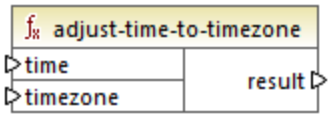
XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Type	Description
time	`xs:time`	La valeur d'entrée de type `xs:time`.

6.6.16.6 adjust-time-to-timezone

Ajuste une valeur `xs:time` dans un fuseau horaire spécifique ou aucun fuseau horaire. Si l'argument **timezone** est une séquence vide, la fonction retourne un `xs:time` sans fuseau horaire. Sinon, il retourne un `xs:time` avec un fuseau horaire.



Langages

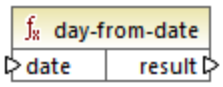
XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Type	Description
time	`xs:time`	La valeur d'entrée de type `xs:time`.
timezone	`xs:dayTimeDuration`	Le fuseau horaire exprimé en tant que valeur `xs:dayTimeDuration`. La valeur peut être négative. Par exemple, une valeur de fuseau horaire de -5 heures peut être exprimée en tant que **`-PT5H`**.

6.6.16.7 day-from-date

Retourne un `xs:integer` représentant la partie day de la valeur `xs:date` fournie en tant qu'argument.



Langages

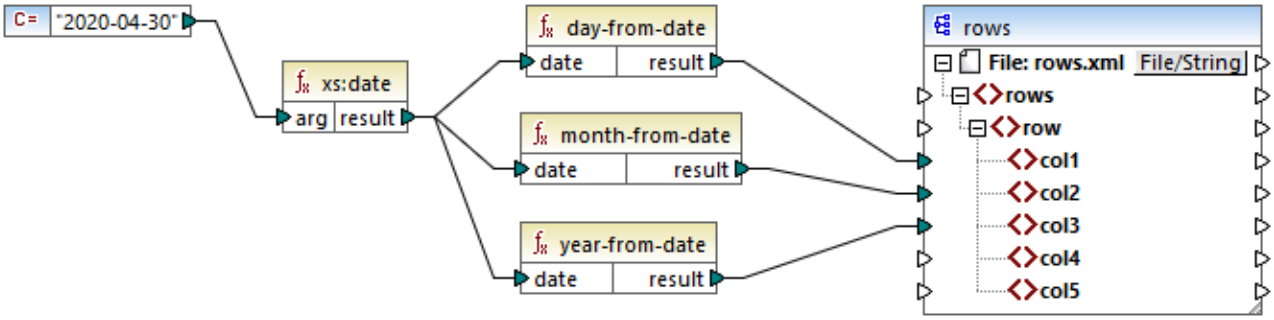
XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Type	Description
date	`xs:date`	La valeur d'entrée de type `xs:date`.

Exemple

Le mappage suivant convertit un string en `xs:date` en utilisant la fonction constructor **`xs:date`**. Les fonctions **`day-from-date`**, **`month-from-date`** et **`year-from-date`** extraient chacune la partie respective de la date et l'écrivent dans un item séparé dans le fichier XML cible.



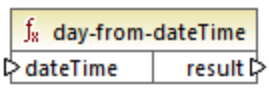
Mappage XQuery 1.0

Le sortie de mappage est la suivante :

```
<rows>
   <row>
      <col1>30</col1>
      <col2>4</col2>
      <col3>2020</col3>
   </row>
</rows>
```

6.6.16.8 day-from-dateTime

Retourne un `xs:integer` représentant la partie day de la valeur `xs:dateTime` fournie en tant qu'argument.



Langages

XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Type	Description
dateTime	`xs:dateTime`	La valeur d'entrée de type `xs:dateTime`.

6.6.16.9 days-from-duration

Retourne un `xs:integer` représentant le composant "days" de la représentation canonique de la valeur de durée fournie en tant qu'argument.

Langages

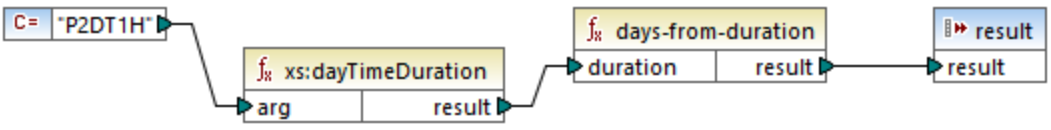
XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Type	Description
duration	`xs:duration`	La valeur d'entrée de type `xs:duration`.

Exemple

Le mappage illustré ci-dessous construit le `xs:dayTimeDuration` de **`P2DT1H`** (2 jours et 1 heure) et le fournit en tant qu'entrée de la fonction **`days-from-duration`**. Le résultat est **2**.



Mappage XSLT 2.0

Note : si la durée est **`P1DT24H`** (1 jour et 24 heures), la fonction retourne **2**, pas **1**. Cela est dû au fait que la représentation canonique de **`P1DT24H`** est en réalité **`P2D`** (2 jours).

6.6.16.10 hours-from-dateTime

Retourne un `xs:integer` représentant la partie hours de la valeur `xs:dateTime` fournie en tant qu'argument.

Langages

XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Type	Description
dateTime	`xs:dateTime`	La valeur d'entrée de type `xs:dateTime`.

6.6.16.11 hours-from-duration

Retourne un `xs:integer` représentant le composant "hours" de la représentation canonique de la valeur de durée fournie en tant qu'argument.

Langages

XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Type	Description
duration	`xs:duration`	La valeur d'entrée de type `xs:duration`.

Exemple

Si la durée est **`PT1H60M`** (1 heure et 60 minutes), la fonction retourne **2**, pas **1**. Cela est dû au fait que la représentation canonique de **`PT1H60M`** est en réalité **`PT2H`** (2 heures).

6.6.16.12 hours-from-time

Retourne un `xs:integer` représentant la partie hours de la valeur `xs:time` fournie en tant qu'argument.

Langages

XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Type	Description
time	`xs:time`	La valeur d'entrée de type `xs:time`.

6.6.16.13 minutes-from-dateTime

Retourne un `xs:integer` représentant la partie minutes de `xs:dateTime` fourni en tant qu'argument.

Langages

XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Type	Description
dateTime	`xs:dateTime`	La valeur d'entrée de type `xs:dateTime`.

6.6.16.14 minutes-from-duration

Retourne un `xs:integer` représentant le composant "minutes" de la représentation canonique de la durée fournie en tant qu'argument.

Langages

XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Type	Description
duration	`xs:duration`	La valeur d'entrée de type `xs:duration`.

Exemple

Si la durée est **`PT1M60S`** (1 minute et 60 secondes), la fonction retourne **2**, pas **1**. Cela est dû au fait que la représentation canonique de **`PT1M60S`** est en réalité **`PT2M`** (2 minutes).

6.6.16.15 minutes-from-time

Retourne un `xs:integer` représentant la partie minutes de la valeur `xs:time` fournie en tant qu'argument.

Langages

XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Type	Description
time	`xs:time`	La valeur d'entrée de type `xs:time`.

6.6.16.16 month-from-date

Retourne un `xs:integer` représentant la partie month de la valeur `xs:date` fournie en tant qu'argument.

Langages

XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Type	Description
date	`xs:date`	La valeur d'entrée de type `xs:date`.

6.6.16.17 month-from-dateTime

Retourne un `xs:integer` représentant la partie month de la valeur `xs:dateTime` fournie en tant qu'argument.

Langages

XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Type	Description
dateTime	`xs:dateTime`	La valeur d'entrée de type `xs:dateTime`.

6.6.16.18 months-from-duration

Retourne un `xs:integer` représentant le composant "months" de la représentation canonique de la valeur de durée fournie en tant qu'argument.

Langages

XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Type	Description
duration	**xs:duration**	La valeur d'entrée de type `xs:duration`.

6.6.16.19 seconds-from-dateTime

Retourne un xs:integer représentant le composant seconds dans la valeur localisée de dateTime.

Langages

XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Type	Description
dateTime	**xs:dateTime**	

6.6.16.20 seconds-from-duration

Retourne un `xs:integer` représentant le composant "seconds" de la représentation canonique de la valeur de durée fournie en tant qu'argument.

Langages

XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Type	Description
duration	**xs:duration**	La valeur d'entrée de type `xs:duration`.

6.6.16.21 seconds-from-time

Retourne un `xs:integer` représentant la partie seconds de la valeur `xs:time` fournie en tant qu'argument.

Langages

XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Type	Description
time	`xs:time`	La valeur d'entrée de type `xs:time`.

6.6.16.22 subtract-dateTimes

Retourne le `xs:dayTimeDuration` qui correspond à la différence entre la valeur normalisée de **dateTime1** et la valeur normalisée de **dateTime2**.

Langages

XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Type	Description
dateTime1	`xs:dateTime`	La première valeur d'entrée.
dateTime2	`xs:dateTime`	La deuxième valeur d'entrée.

6.6.16.23 subtract-dates

Retourne le `xs:dayTimeDuration` qui correspond à la différence entre la valeur normalisée de **date1** et la valeur normalisée de **date2**.

Langages

XQuery, XSLT 2.0, XSLT 3.0.

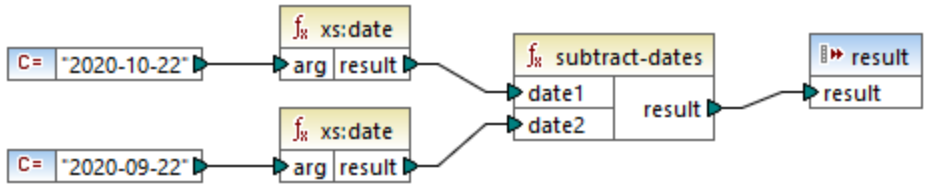
Paramètres

Nom	Type	Description
date1	`xs:date`	La première valeur d'entrée.

Nom	Type	Description
date2	`xs:date`	La deuxième valeur d'entrée.

Exemple

Le mappage illustré ci-dessous soustrait deux dates (2020-10-22 moins 2020-09-22). Le résultat est la valeur **`P30D`** de type `xs:dayTimeDuration`, qui représente une durée de 30 jours.



6.6.16.24 subtract-times

Retourne le `xs:dayTimeDuration` qui correspond à la différence entre la valeur normalisée de **time1** et la valeur normalisée de **time2**.

Langages

XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Type	Description
time1	`xs:time`	La première valeur d'entrée.
time2	`xs:time`	La deuxième valeur d'entrée.

6.6.16.25 timezone-from-date

Retourne le composant timezone de la date fournie en tant qu'argument. Le résultat est un `xs:dayTimeDuration` qui indique une déviation depuis UTC; sa valeur peut s'étendre de +14:00 à -14:00 heures, inclus tous deux.

Langages

XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Type	Description
date	`xs:date`	La valeur d'entrée de type `xs:date`.

6.6.16.26 timezone-from-dateTime

Retourne le composant timezone de la valeur `xs:dateTime` fournie en tant qu'argument. Le résultat est un `xs:dayTimeDuration` qui indique une déviation depuis UTC; sa valeur peut s'étendre de +14:00 à -14:00 heures, inclus tous deux.

Langages

XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Type	Description
dateTime	`xs:dateTime`	La valeur d'entrée de type `xs:dateTime`.

6.6.16.27 timezone-from-time

Retourne le composant timezone de la valeur `xs:time` fournie en tant qu'argument. Le résultat est un `xs:dayTimeDuration` qui indique une déviation depuis UTC; sa valeur peut s'étendre de +14:00 à -14:00 heures, inclus tous deux.

Langages

XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Type	Description
time	`xs:time`	La valeur d'entrée de type `xs:time`.

6.6.16.28 year-from-date

Retourne un `xs:integer` représentant la partie year de la valeur `xs:date` fournie en tant qu'argument.

Langages

XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Type	Description
date	**xs:date**	La valeur d'entrée de type `xs:date`.

6.6.16.29 year-from-dateTime

Retourne un `xs:integer` représentant la partie year de la valeur `xs:dateTime` fournie en tant qu'argument.

Langages

XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Type	Description
dateTime	**xs:dateTime**	La valeur d'entrée de type `xs:dateTime`.

6.6.16.30 years-from-duration

Retourne un `xs:integer` représentant le composant "years" de la représentation canonique de la valeur de durée fournie en tant qu'argument.

Langages

XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Type	Description
duration	**xs:duration**	La valeur d'entrée de type `xs:duration`.

6.6.17 xpath2 | node functions

Les fonctions de nœud provenant de la bibliothèque **xpath2** fournissent des informations concernant les nœuds (items) dans un composant de mappage.

La fonction **lang** prend un argument string qui identifie un code de langage (comme par exemple "en"). La fonction retourne **true** ou **false** selon le fait que le nœud contextuel a un attribut `xml:lang` avec une valeur qui correspond à l'argument de la fonction.

Les fonctions **local-name**, **name** et **namespace-uri**, retournent, respectivement, le nom local, le nom et l'URI d'espace de nom du nœud d'entrée. Par exemple, pour le nœud **altova:Products**, le nom local est **Products**, le nom est **altova:Products** et l'URI de l'espace de noms est l'URI de l'espace de noms auquel le préfixe altova: est lié (voir l'exemple donné pour la fonction **local-name** 344). Chacune de ces trois fonctions a deux variantes :

- Avec aucun argument : la fonction est ensuite appliquée au nœud contextuel (pour un exemple d'un nœud contextuel, voir l'exemple donné pour la fonction **lang** 342).
- Avec un argument qui doit être un nœud : la fonction est appliquée au nœud contextuel.

La fonction **number** prend un nœud en tant qu'entrée, atomise le nœud (c'est à dire extrait son contenu) et convertit la valeur en une décimale puis retourne la valeur convertie. Il existe deux variantes de la fonction **number** :

- Avec aucun argument : la fonction est ensuite appliquée au nœud contextuel (pour un exemple d'un nœud contextuel, voir l'exemple donné pour la fonction **lang** 342).
- Avec un argument qui doit être un nœud : la fonction est appliquée au nœud contextuel.

6.6.17.1 lang

Retourne **true** si le nœud contextuel a un attribut `xml:lang` avec une valeur soit qui correspond exactement à l'argument **testlang**, ou en est un sous-ensemble. Sinon, la fonction retourne **false**.

Langages

XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

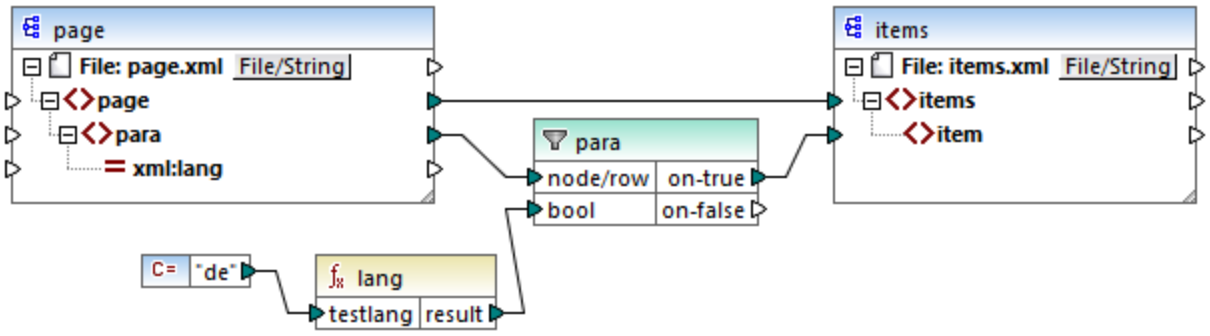
Nom	Type	Description
testlang	**xs:string**	Le code de langage à vérifier, par exemple, "en".

Exemple

Le XML suivant contient des éléments **para** avec des valeurs différentes pour l'attribut `xml:lang`.

```
<page>
   <para xml:lang="en">Good day!</para>
   <para xml:lang="fr">Bonjour!</para>
   <para xml:lang="de-AT">Grüss Gott!</para>
   <para xml:lang="de-DE">Guten Tag!</para>
   <para xml:lang="de-CH">Grüezi!</para>
</page>
```

Le mappage illustré ci-dessous filtre uniquement les paragraphes en allemand, quelle que soit la variante du pays, avec l'aide de la fonction **lang**.



Mappage XSLT 2.0

Dans le mappage ci-dessus, pour chaque **para** dans la source, un **item** est créé dans la cible, de manière conditionnelle. La condition est fournie par un filtre qui transfère à la cible uniquement les nœuds dans lesquels la fonction **lang** retourne **true**. Ainsi, seul les nœuds qui ont l'attribut `xml:lang` définis sur "de" (ou un sous-ensemble de "de") satisferont la condition du filtre. Par conséquent, la sortie de mappage est le suivant :

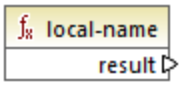
```
<items>
   <item>Grüss Gott!</item>
   <item>Guten Tag!</item>
   <item>Grüezi!</item>
</items>
```

Veuillez noter que la fonction **lang** opère dans le contexte de chaque **para**, à cause de la connexion parent entre **para** et **item**, voir aussi Le contexte de mappage 407.

6.6.17.2 local-name

Retourne la partie locale du nom du nœud contextuel en tant qu'un `xs:string`. Il s'agit d'une variante sans paramètres de la fonction **local-name** où le nœud contextuel est déterminé par les connexions dans votre

mappage. Pour spécifier un nœud explicitement, utiliser la fonction **local-name** 344 qui prend un nœud d'entrée en tant que paramètre.

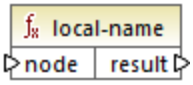


Langages

XQuery, XSLT 2.0, XSLT 3.0.

6.6.17.3 local-name

Retourne la partie locale du nom du **nœud** contextuel en tant qu'un `xs:string`.



Langages

XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

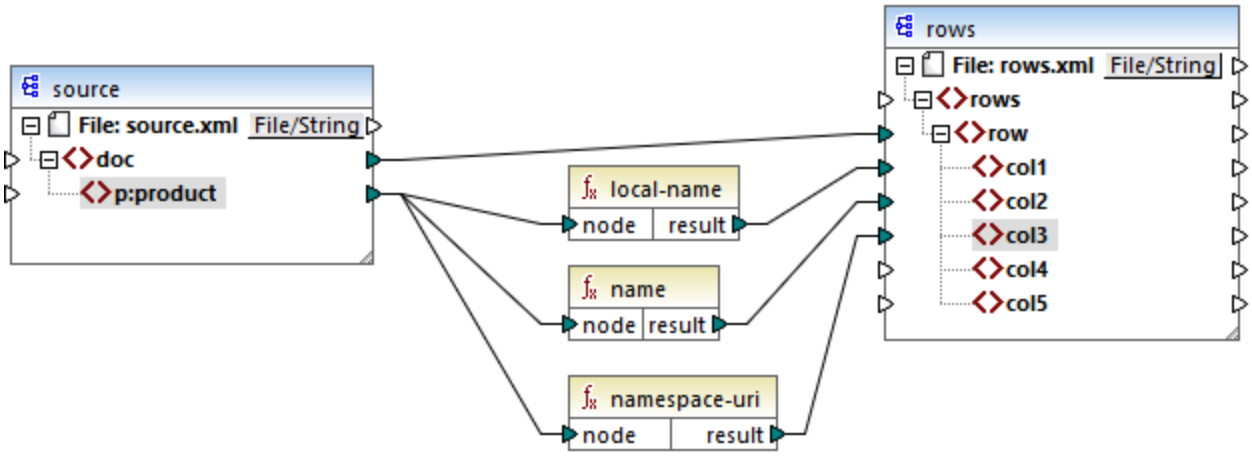
Nom	Type	Description
node	**node()**	Le nœud d'entrée.

Exemple

Dans le fichier XML suivant, le nom de l´élément `p:product` est un nom qualifié préfixé (QName). Le préfixe "p" est mappé dans l'espace de noms "http://mycompany.com".

```
<?xml version="1.0" encoding="UTF-8"?>
<doc xmlns:p="http://mycompany.com" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="source.xsd">
   <p:product/>
</doc>
```

Le mappage suivant extrait le nom local, le nom et l'URI d'espace de noms du nœud et écrit ces valeurs dans un fichier cible :



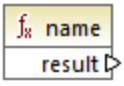
Mappage XSLT 2.0

La sortie de mappage est affichée ci-dessous. Chaque item **col** liste le résultat des fonctions **local-name**, **name** et **namespace-uri**, respectivement.

```
<rows>
   <row>
      <col1>product</col1>
      <col2>p:product</col2>
      <col3>http://mycompany.com</col3>
   </row>
</rows>
```

6.6.17.4 name

Retourne le nom du nœud contextuel. Il s'agit d'une variante sans paramètres de la fonction **name** où le nœud contextuel est déterminé par les connexions dans votre mappage. Pour spécifier un nœud explicitement, utiliser la fonction **name** 346 qui prend un nœud d'entrée en tant que paramètre.

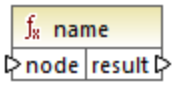


Langages

XQuery, XSLT 2.0, XSLT 3.0.

6.6.17.5 name

Retourne le nom d'un nœud.



Langages

XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Type	Description
node	`node()`	Le nœud d'entrée.

Exemple

Voir l'exemple donné pour la fonction **local-name** 344.

6.6.17.6 namespace-uri

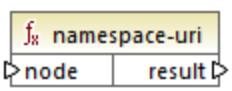
Retourne l'espace de nom URI du QName du nœud contextuel en tant `xs:string`. Il s'agit d'une variante sans paramètres de la fonction **namespace-uri** où le nœud contextuel est déterminé par les connexions dans votre mappage. Pour spécifier un nœud explicitement, utiliser la fonction **namespaces- uri** 346 qui prend un nœud d'entrée en tant que paramètre.

Langages

XQuery, XSLT 2.0, XSLT 3.0.

6.6.17.7 namespace-uri

Retourne l'espace de nom URI du QName du **nœud**, en tant que `xs:string`.



Langages
XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

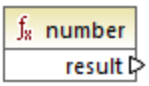
Nom	Type	Description
node	`node()`	Le nœud d'entrée.

Exemple
Voir l'exemple donné pour la fonction **local-name** 344.

6.6.17.8 number

Retourne la valeur du nœud contextuel, converti dans un `xs:double`. Il s'agit d'une variante sans paramètres de la fonction **number** où le nœud contextuel est déterminé par les connexions dans votre mappage. Pour spécifier un nœud explicitement, utiliser la fonction **number** 347 qui prend un nœud d'entrée en tant que paramètre.

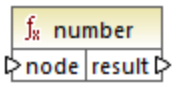
Les seuls types qui peuvent être convertis dans des nombres sont des booléennes, des strings numériques et d'autres types numériques. Les valeurs d'entrée non-numériques (comme un string non-numérique) résultent en NaN (Not a Number - Pas un Nombre).



Langages
XQuery, XSLT 2.0, XSLT 3.0.

6.6.17.9 number

Retourne la valeur du **nœud**, converti dans un `xs:double`. Les seuls types qui peuvent être convertis dans des nombres sont des booléennes, des strings numériques et d'autres types numériques. Les valeurs d'entrée non-numériques (comme un string non-numérique) résultent en NaN (Not a Number - Pas un Nombre).



Langages
XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

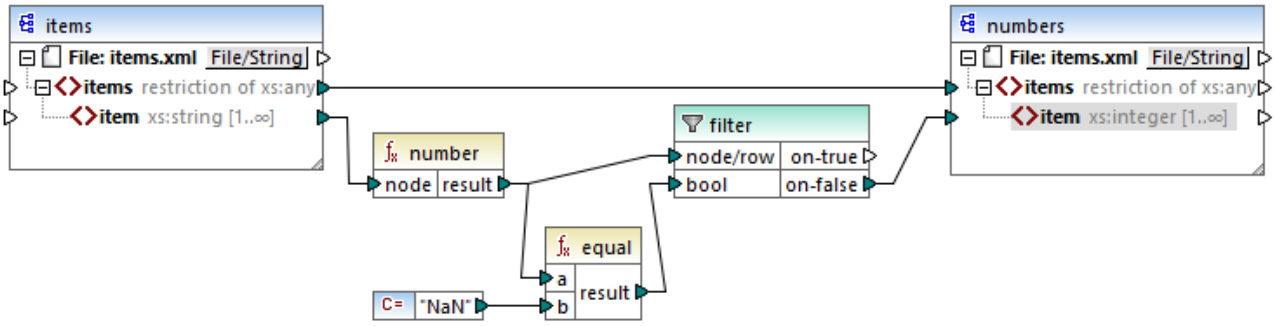
Nom	Type	Description
node	`mf:atomic`	Le nœud d'entrée.

Exemple

Le XML suivant contient des items de type `string`:

```
<items>
   <item>1</item>
   <item>2</item>
   <item>Jingle Bells</item>
</items>
```

Le mappage illustré ci-dessous tente de convertir tous ces strings en des valeurs numériques et les écrit dans un fichier XML cible. Veuillez noter que le type de données de **item** dans le composant XML cible est `xs:integer` alors que l'**item** de source est de type de données `xs:string`. Si la conversion échoue, l'item doit être sauté et ne sera pas copié dans le fichier cible.



Mappage XSLT 2.0

Un filtre a été utilisé pour parvenir à l'objectif de mappage. La fonction **`equal`** vérifie si le résultat de la conversion est "NaN". Si cela est faux, cela indique une conversion réussie, l'item est donc copié dans la cible. La sortie du mappage est comme suit :

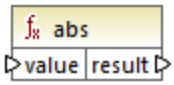
```
<items>
   <item>1</item>
   <item>2</item>
</items>
```

6.6.18 xpath2 | numeric functions

Les fonctions numériques de la bibliothèque **xpath2** comprennent les fonctions **`abs`** et **`round-half-to-even`**.

6.6.18.1 abs

Retourne la valeur absolue de l'argument. Par exemple, si l'argument d'entrée est **-2** ou **2**, la fonction retourne **2**.



Langages

XQuery, XSLT 2.0, XSLT 3.0.

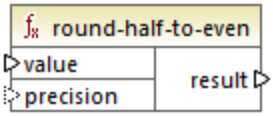
Paramètres

Nom	Type	Description
value	`xs:decimal`	La valeur d'entrée.

6.6.18.2 round-half-to-even

La fonction `round-half-to-even` arrondit le nombre fourni (premier argument) à la précision décimale (nombre de chiffres après la virgule) fourni dans le second argument optionnel. Par exemple, si le premier argument est **2.141567** et que le second argument est **3**, alors le premier argument (le nombre) est arrondi à trois chiffres après la virgule, le résultat sera donc **2.142**. Si aucune précision décimale (second argument) n'est fournie, le nombre sera arrondi à zéro places décimales, c'est à dire donc à un entier.

Le terme "even" dans le nom de la fonction réfère à l'arrondissement à un nombre pair lorsqu'un chiffre dans le nombre fourni se trouve entre deux valeurs. Par exemple, `round-half-to-even(3.475, 2)` retournerait **3.48**.



Langages

XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Type	Description
valeur	`xs:decimal`	Argument obligatoire qui fournit la valeur d'entrée à arrondir.

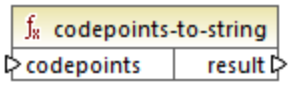
Nom	Type	Description
precision	`xs:integer`	Argument optionnel qui spécifie le nombre de décimales à arrondir. La valeur par défaut est de **0**.

6.6.19 xpath2 | string functions

Les fonctions string de la bibliothèque **xpath2** vous permettent de traiter des strings (cela inclut la comparaison de strings, la conversion de strings en casse majuscule ou minuscule, l'extraction de sous-strings depuis des strings, etc.).

6.6.19.1 codepoints-to-string

Crée un string depuis une séquence de points de code Unicode. Cette fonction est le contraire de la fonction `string-to-codepoints` 358.



Langages

XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Type	Description
codepoints	`ZeroOrMore xs:integer`	Cette entrée doit être connectée à une séquence d'items de type integer, où chaque entier spécifie un point de code Unicode.

Exemple

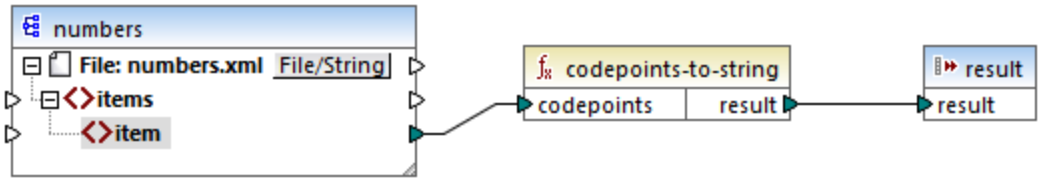
L'XML suivant contient plusieurs éléments **item** qui stockent des valeurs de point de code chacun Unicode.

```
<items>
   <item>77</item>
   <item>97</item>
   <item>112</item>
   <item>70</item>
   <item>111</item>
   <item>114</item>
   <item>99</item>
```



```
   <item>101</item>
</items>
```

Le mappage illustré ci-dessous fournit la séquence des items en tant qu'argument dans la fonction **codepoint-to-string**.



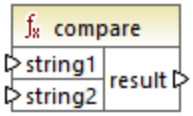
Mappage XSLT 2.0

La sortie de mappage est **MapForce**.

6.6.19.2 compare

La fonction **compare** prend deux strings en tant qu'arguments et les compare en terme d'égalité et alphabétiquement. Si **string1** est alphabétiquement inférieur à **string2** (par exemple les deux strings sont "A" et "B"), la fonction retourne **-1**. Si les deux strings sont égaux (par exemple, "A" et "A"), la fonction retourne **0**. Si **string1** est supérieur à **string2** (par exemple, "B" et "A"), alors la fonction retourne **1**.

Cette variante de la fonction utilise la collation par défaut, qui est Unicode. Une autre variante 352 de cette fonction existe là où vous pouvez fournir la collation en tant qu'argument.



Langages

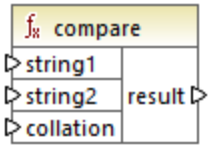
XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Type	Description
string1	xs:string	Le premier string d'entrée.
string2	xs:string	Le second string d'entrée.

6.6.19.3 compare

La fonction **compare** prend deux strings en tant qu'arguments et les compare en terme d'égalité et alphabétiquement, en utilisant la collation fournie en tant qu'argument. Si **string1** est alphabétiquement inférieur à **string2** (par exemple les deux strings sont "A" et "B"), la fonction retourne **-1**. Si les deux strings sont égaux (par exemple, "A" et "A"), la fonction retourne **0**. Si **string1** est supérieur à **string2** (par exemple, "B" et "A"), alors la fonction retourne **1**.



Langages

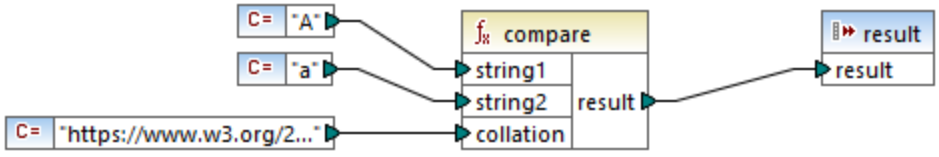
XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Type	Description
string1	`xs:string`	Le premier string d'entrée.
string2	`xs:string`	Le second string d'entrée.
collation	`xs:string`	Spécifie la collation à utiliser pour la comparaison de string. Cette entrée peut provenir de la sortie de la fonction **default-collation**[323] ou il peut s'agir d'une collation comme `http://www.w3.org/2005/xpath-functions/collation/html-ascii-case-insensitive`.

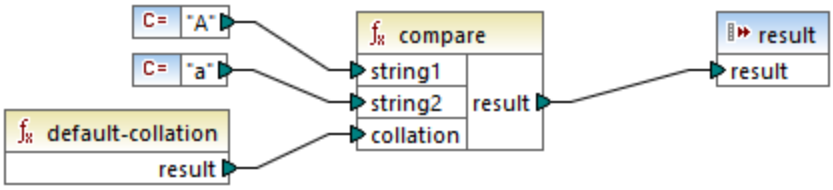
Exemple

Le mappage suivant compare les strings "A" et "a" en utilisant la collation insensible à la casse `http://www.w3.org/2005/xpath-functions/collation/html-ascii-case-insensitive`, qui est fournie par une constante.



Mappage XSLT 2.0

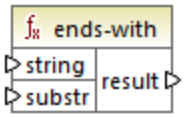
Le résultat du mappage ci-dessus est **0**, signifiant que les deux strings sont traités à égalité. Néanmoins, si vous remplacez la collation avec celle fournie par la fonction **default-collation**, la collation change pour passer à la collation de point de code Unicode par défaut, et le résultat de mappage devient **-1** ("A" est alphabétiquement inférieur à "a").



6.6.19.4 ends-with

Retourne **true** si **string** se termine avec **substr**; **false** sinon. La valeur retournée est de type `xs:boolean`.

Cette variante de la fonction utilise la collation par défaut, qui est Unicode. Une autre variante[354] de cette fonction existe là où vous pouvez fournir la collation en tant qu'argument.



Langages

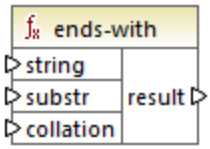
XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Type	Description
string	`xs:string`	Le string d'entrée (c'est à dire la "pile de foin").
substr	`xs:string`	Le sous-string (c'est à dire l'"aiguille").

6.6.19.5 ends-with

Retourne **true** si **string** se termine avec **substr**; **false** sinon. La valeur retournée est de type `xs:boolean`.



Langages

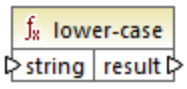
XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Type	Description
string	`xs:string`	Le string d'entrée (c'est à dire la "pile de foin").
substr	`xs:string`	Le sous-string (c'est à dire l'"aiguille").
collation	`xs:string`	Spécifie la collation à utiliser pour la comparaison de string. Cette entrée peut provenir de la sortie de la fonction **`default-collation`** 323 ou il peut s'agir d'une collation comme `http://www.w3.org/2005/xpath-functions/collation/html-ascii-case-insensitive`.

6.6.19.6 lower-case

Retourne la valeur de **string** après avoir traduit chaque caractère dans son correspondant en minuscule.



Langages

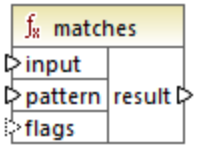
XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Type	Description
string	`xs:string`	La valeur d'entrée.

6.6.19.7 matches

La fonction **`matches`** teste si un string fourni (le premier argument) correspond à une expression régulière (le second argument). La syntaxe des expressions régulières doit être celui défini pour la facette `pattern` du Schéma XML. La fonction retourne **true** si le string correspond à l'expression régulière, **false** sinon.



Langages

XQuery, XSLT 2.0, XSLT 3.0.

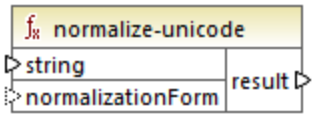
Paramètres

Nom	Type	Description
input	`xs:string`	Type
pattern	`xs:string`	L'expression régulière à faire correspondre, voir Expressions régulière[228].
flags	`xs:string`	Argument optionnel influe sur la correspondance. Cet argument peut fournir n'importe quelle combinaison des flags suivants : `i`, `m`, `s`, `x`. Plusieurs flags peuvent être utilisés, par exemple, `imx`. Si aucun flag n'est utilisé, les valeurs par défaut des quatre flags seront utilisées. Les quatre flags sont les suivants : `i` Utiliser le mode insensible à la casse. Le défaut est sensible à la casse. `m` Utiliser le mode multiligne, dans lequel le string d'entrée est considéré avoir plusieurs lignes, chacune séparée par un caractère newline (`x0a`). Les caractères meta **^** et **$** indiquent le début et la fin de chaque ligne. Le mode par défaut est le mode string, dans lequel le string commence et termine par les caractères méta **^** et **$**.

Nom	Type	Description
		s Utiliser le mode dot-all. Le mode par défaut est le mode not-dot-all, dans lequel le caractére méta . Correspond à tous les caractères sauf le caractère newline (x0a). Dans le mode dot-all, le point correspond aussi au caractère newline. x Ignorer l'espace blanc. Par défaut, les caractères d'espace blanc ne sont pas ignorés.

6.6.19.8 normalize-unicode

Retourne la valeur de **string** normalisée conformément aux règles du formulaire de normalisation spécifié (le second argument). Pour plus d'informations concernant la normalisation Unicode, voir §2.2 de https://www.w3.org/TR/charmod-norm/.



Langages

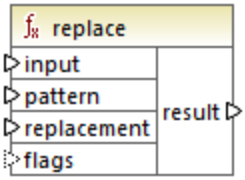
XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Type	Description
string	xs:string	La valeur string à être normalisé.
normalizationForm	xs:string	L'argument optionnel fournit le formulaire de normalisation. Le défaut est Unicode Normalization Form C (NFC). Les formulaires de normalisation NFC, NFD, NFKC et NFKD sont pris en charge.

6.6.19.9 replace

Cette fonction prend un string d'entrée, une expression régulière et un string de remplacement en tant qu'arguments. Elle remplace toutes les correspondances de l'expression régulière dans le string d'entrée avec le string de remplacement.. Si l'expression régulière correspond à deux strings se chevauchant dans le string d'entrée, seule la première correspondance est remplacée.



Langages

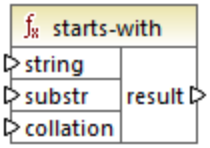
XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Type	Description
input	`xs:string`	Type
pattern	`xs:string`	L'expression régulière à faire correspondre, voir Expressions régulière 228.
replacement	`xs:string`	Le string de remplacement.
flags	`xs:string`	Argument optionnel influe sur la correspondance. Cet argument est utilisé de la même manière que l'argument **flags** de la fonction `matches` 355.

6.6.19.10 starts-with

Retourne **true** si **string** commence avec **substr**; **false** sinon. La valeur retournée est de type `xs:boolean`. La comparaison de string prend place conformément à la collation spécifiée.



Langages

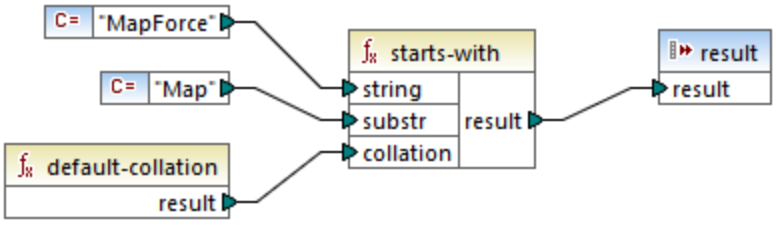
XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Type	Description
string	`xs:string`	Le string d'entrée (c'est à dire la "pile de foin").
substr	`xs:string`	Le sous-string (c'est à dire l'"aiguille").
collation	`xs:string`	Spécifie la collation à utiliser pour la comparaison de string. Cette entrée peut provenir de la sortie de la fonction **`default-collation`** 323 ou il peut s'agir d'une collation comme `http://www.w3.org/2005/xpath-functions/collation/html-ascii-case-insensitive`.

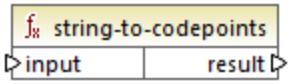
Exemple

Le mappage suivant retourne la valeur **`true`**, parce que le string d'entrée "MapForce" commence avec le sous-string "Map", en partant du principe que la collation Unicode par défaut est utilisée.



6.6.19.11 string-to-codepoints

Retourne la séquence des points de code Unicode (valeur d'entier) qui constitue le string fourni en tant qu'argument. Cette fonction est le contraire de la fonction **`codepoints-to-string`** 350.



Langages

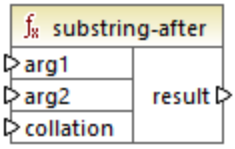
XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Type	Description
input	`xs:string`	Le string d'entrée

6.6.19.12 substring-after

Retourne la partie du string **arg1** qui se produit après le string **arg2**.



Langages

XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

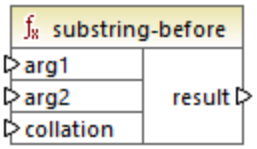
Nom	Type	Description
arg1	`xs:string`	Le string d'entrée (c'est à dire la "pile de foin").
arg2	`xs:string`	Le sous-string (c'est à dire l'"aiguille").
collation	`xs:string`	Spécifie la collation à utiliser pour la comparaison de string. Cette entrée peut provenir de la sortie de la fonction **`default-collation`** 323 ou il peut s'agir d'une collation comme `http://www.w3.org/2005/xpath-functions/collation/html-ascii-case-insensitive`.

Exemple

Si **arg1** est "MapForce", **arg2** est "Map", et **collation** est **`default-collation`** 323, la fonction retourne "Force".

6.6.19.13 substring-before

Retourne la partie du string **arg1** qui se produit avant le string **arg2**.



Langages

XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

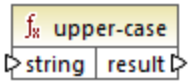
Nom	Type	Description
arg1	`xs:string`	Le string d'entrée (c'est à dire la "pile de foin").
arg2	`xs:string`	Le sous-string (c'est à dire l'"aiguille").
collation	`xs:string`	Spécifie la collation à utiliser pour la comparaison de string. Cette entrée peut provenir de la sortie de la fonction **`default-collation`** 323 ou il peut s'agir d'une collation comme `http://www.w3.org/2005/xpath-functions/collation/html-ascii-case-insensitive`.

Exemple

Si **arg1** est "MapForce", **arg2** est "Force", et **collation** est **`default-collation`** 323, la fonction retourne "Map".

6.6.19.14 upper-case

Retourne la valeur de **string** après avoir traduit chaque caractère dans son correspondant en majuscule.



Langages
XQuery, XSLT 2.0, XSLT 3.0.

Paramètres

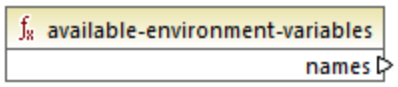
Nom	Type	Description
string	`xs:string`	Type

6.6.20 xpath3 | external information functions

Les fonctions d'information externe de la librairie **xpath3** vous permettent d'obtenir l'information sur l'environnement d'exécution XSLT ou d'extraire des données de ressources externes.

6.6.20.1 available-environment-variables

Retourne une liste de noms de variables d'environnement qui conviennent pour passer à la fonction `environment-variable`, comme séquence de strings (éventuellement vide).



Langages
XSLT 3.0.

6.6.20.2 environment-variable

Retourne la valeur d'une variable d'environnement système, si elle existe. The type de retour est `xs:string`.

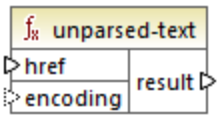
Langages
XSLT 3.0.

Paramètres

Nom	Type	Description
nom	`xs:string`	Le nom de la varaible d'environnement.

6.6.20.3 unparsed-text

Lit une ressource externe (par exemple, un fichier) et retourne une représentation string de la ressource.



Langages

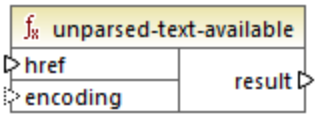
XSLT 3.0.

Paramètres

Nom	Type	Description
href	`xs:string`	Un string sous forme d'une référence URI.
encoding	`xs:string`	Argument optionnel. Spécifie le nom de l'encodage, par exemple "UTF-8", "UTF-16". Si l'encodage ne peut pas être déterminé automatiquement, alors UTF-8 est présumé.

6.6.20.4 unparsed-text-available

Détermine si un appel vers **`unparsed-text`** avec des arguments particuliers réussirait. Le type de retour est `xs:boolean`.



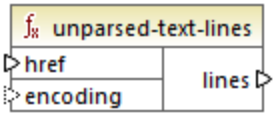
Langages
XSLT 3.0.

Paramètres

Nom	Type	Description
href	`xs:string`	Un string sous forme d'une référence URI.
encoding	`xs:string`	Argument optionnel. Spécifie le nom de l'encodage, par exemple "UTF-8", "UTF-16". Si l'encodage ne peut pas être déterminé automatiquement, alors UTF-8 est présumé.

6.6.20.5 unparsed-text-lines

Lit une ressource externe (par exemple, un fichier) et retourne ses contenus comme une séquence de strings, une par ligne de texte dans la représentation string de la ressource.



Langages
XSLT 3.0.

Paramètres

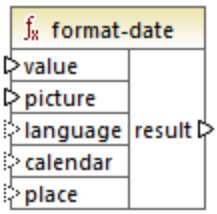
Nom	Type	Description
href	`xs:string`	Un string sous forme d'une référence URI.
encoding	`xs:string`	Argument optionnel. Spécifie le nom de l'encodage, par exemple "UTF-8", "UTF-16". Si l'encodage ne peut pas être déterminé automatiquement, alors UTF-8 est présumé.

6.6.21 xpath3 | formatting functions

Les fonctions de formatage disponibles de la bibliothèque **xpath3** sont utilisées pour formater la date, l'heure et les valeurs d'entier.

6.6.21.1 format-date

Retourne un string contenant une `xs: date` valeur formatée pour l'affichage.



Langages

XSLT 3.0.

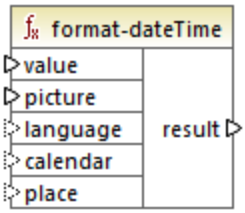
Paramètres

Nom	Type	Description
value	`xs:date`	Fournit la valeur `xs: date` à formater. Paramètre obligatoire.
image	`xs:string`	Paramètre obligatoire. Voir section 9.8.4.1 de la Recommandation W3C « XPath and XQuery Functions and Operators 3.1 » (https://www.w3.org/TR/xpath-functions-31).
language	`xs:string`	Paramètre optionnel. Voir section 9.8.4.8 de la Recommandation W3C « XPath and XQuery Functions and Operators 3.1 » (https://www.w3.org/TR/xpath-functions-31).
calendrier	`xs:string`	Comme ci-dessus.

Nom	Type	Description
place	`xs:string`	Comme ci-dessus.

6.6.21.2 format-dateTime

Retourne un string contenant une `xs: dateTime` valeur formatée pour l'affichage.



Langages

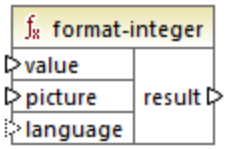
XSLT 3.0.

Paramètres

Nom	Type	Description
value	`xs:dateTime`	Fournit la valeur `xs: dateTime` à formater.
image	`xs:string`	Paramètre obligatoire. Voir section 9.8.4.1 de la Recommandation W3C « XPath and XQuery Functions and Operators 3.1 » (https://www.w3.org/TR/xpath-functions-31).
language	`xs:string`	Paramètre optionnel. Voir section 9.8.4.8 de la Recommandation W3C « XPath and XQuery Functions and Operators 3.1 » (https://www.w3.org/TR/xpath-functions-31).
calendrier	`xs:string`	Comme ci-dessus.
place	`xs:string`	Comme ci-dessus.

6.6.21.3 format-integer

Formate un entier conformément à un string d'image donné, utilisant les conventions d'un langage naturel donné, si spécifié.



Langages

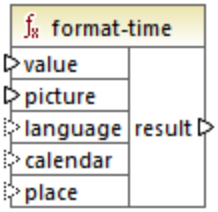
XSLT 3.0.

Paramètres

Nom	Type	Description
value	`xs:integer`	La valeur de l'entier de l'entrée à formater.
image	`xs:string`	Paramètre obligatoire. Voir la section 4.6.1 de la Recommandation W3C « XPath and XQuery Functions and Operators 3.1 » (https://www.w3.org/TR/xpath-functions-31).
language	`xs:string`	Paramètre optionnel. Spécifie le langage naturel selon la manière avec laquelle la valeur doit être formatée. Si spécifié, la valeur doit soit être un string vide ou toute valeur autorisée pour l'attribut `xml:lang` conformément à la Recommandation W3C « Extensible Markup Language (XML) 1.0 » (https://www.w3.org/TR/xml).

6.6.21.4 format-time

Retourne un string contenant une `xs:time` valeur formatée pour l'affichage.



Langages

XSLT 3.0.

Paramètres

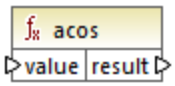
Nom	Type	Description
value	`xs:time`	Fournit la valeur `xs:time` à formater.
image	`xs:string`	Paramètre obligatoire. Voir section 9.8.4.1 de la Recommandation W3C « XPath and XQuery Functions and Operators 3.1 » (https://www.w3.org/TR/xpath-functions-31).
language	`xs:string`	Paramètre optionnel. Voir section 9.8.4.8 de la Recommandation W3C « XPath and XQuery Functions and Operators 3.1 » (https://www.w3.org/TR/xpath-functions-31).
calendrier	`xs:string`	Comme ci-dessus.
place	`xs:string`	Comme ci-dessus.

6.6.22 xpath3 | math functions

Les fonctions mathématiques de la bibliothèque **xpath3** sont utilisées pour effectuer des calculs trigonométriques ou autres calculs mathématiques.

6.6.22.1 acos

Retourne l'arc cosinus d'un angle, allant de **0** à **pi**.



Langages

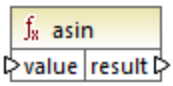
XSLT 3.0.

Paramètres

Nom	Type	Description
value	`xs:double`	La valeur d'entrée.

6.6.22.2 asin

Retourne l'arc sinus d'un angle, allant de **-pi/2** à **pi/2**.



Langages

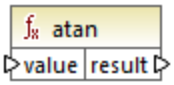
XSLT 3.0.

Paramètres

Nom	Type	Description
value	`xs:double`	La valeur d'entrée.

6.6.22.3 atan

Retourne l'arc tangente d'un angle, allant de **-pi/2** à **pi/2**.



Langages

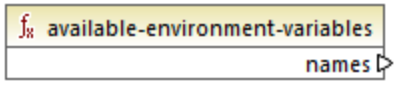
XSLT 3.0.

Paramètres

Nom	Type	Description
value	`xs:double`	La valeur d'entrée.

6.6.22.4 atan2

Retourne une liste de noms de variables d'environnement qui conviennent pour passer à la fonction `environment-variable`, comme séquence de strings (éventuellement vide).

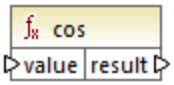


Langages

XSLT 3.0.

6.6.22.5 cos

Retourne le cosinus trigonométrique de l'angle indiqué par la valeur. L'unité de la valeur est radian.



Langages

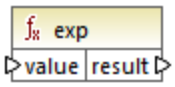
XSLT 3.0.

Paramètres

Nom	Type	Description
value	`xs:double`	La valeur d'entrée.

6.6.22.6 exp

Retourne le nombre d'Euler *e* à la puissance de la valeur.



Langages

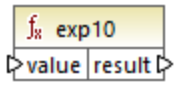
XSLT 3.0.

Paramètres

Nom	Type	Description
value	`xs:double`	La valeur d'entrée.

6.6.22.7 exp10

Retourne 10 à la puissance de la valeur.



Langages

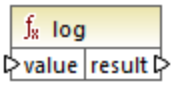
XSLT 3.0.

Paramètres

Nom	Type	Description
value	`xs:double`	La valeur d'entrée.

6.6.22.8 log

Retourne le logarithme naturel (base e) d'une valeur.



Langages

XSLT 3.0.

Paramètres

Nom	Type	Description
value	`xs:double`	La valeur d'entrée.

6.6.22.9 log10

Retourne le logarithme décimal (base 10) d'une valeur.

Langages

XSLT 3.0.

Paramètres

Nom	Type	Description
value	`xs:double`	La valeur d'entrée.

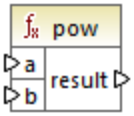
6.6.22.10 pi

Retourne une approximation vers la constante mathématique **pi**.

Langages
XSLT 3.0.

6.6.22.11 pow

Retourne la valeur de **a** élevé à la puissance de **b**.



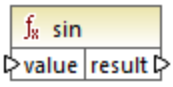
Langages
XSLT 3.0.

Paramètres

Nom	Type	Description
a	`xs:double`	La valeur d'entrée **a**.
b	`xs:double`	La valeur d'entrée **b**.

6.6.22.12 sin

Retourne le sinus trigonométrique d'un angle indiqué par la valeur. L'unité de valeur est en radian.



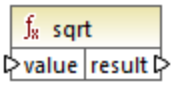
Langages
XSLT 3.0.

Paramètres

Nom	Type	Description
value	`xs:double`	La valeur d'entrée.

6.6.22.13 sqrt

Retourne la racine carrée non négative de l'argument.



Langages

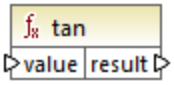
XSLT 3.0.

Paramètres

Nom	Type	Description
value	`xs:double`	La valeur d'entrée.

6.6.22.14 tan

Retourne la tangente trigonométrique de l'angle indiquée par la valeur. L'unité de valeur est en radian.



Langages

XSLT 3.0.

Paramètres

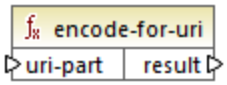
Nom	Type	Description
value	`xs:double`	La valeur d'entrée.

6.6.23 xpath3 | URI functions

Les fonctions URI de la bibliothèque **xpath3** réalisent l'encodage, l'échappement et la conversion de valeurs pour l'utilisation dans les URI.

6.6.23.1 encode-for-uri

Encode les caractères réservés dans un string qui est identifié à être utilisé dans le segment de chemin d'un URI. Pour plus d'information concernant cette fonction, voir la section 6.2 de la Recommandation W3C de « XPath and XQuery Functions and Operators 3.1 » (https://www.w3.org/TR/xpath-functions-31).



Langages

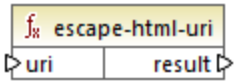
XSLT 3.0.

Paramètres

Nom	Type	Description
uri-part	`xs:string`	La valeur d'entrée de l'URI à encoder.

6.6.23.2 escape-html-uri

Échappe un URI de la même manière que les agents utilisateurs d'HTML gèrent des valeurs d'attribut prévues contenir des URI. Pour plus d'information concernant cette fonction, voir la section 6.4 de la Recommandation W3C de « XPath and XQuery Functions and Operators 3.1 » (https://www.w3.org/TR/xpath-functions-31).



Langages

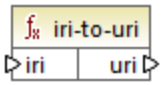
XSLT 3.0.

Paramètres

Nom	Type	Description
uri	`xs:string`	La valeur d'entrée de l'URI à échapper.

6.6.23.3 iri-to-uri

Convertit un string contenant l'IRI (Internationalized Resource Identifier) en un URI (Uniform Resource Identifier). Pour plus d'information concernant cette fonction, voir la section 6.3 de la Recommandation W3C de « XPath and XQuery Functions and Operators 3.1 » (https://www.w3.org/TR/xpath-functions-31).



Langages

XSLT 3.0.

Paramètres

Nom	Type	Description
iri	`xs:string`	La valeur d'entrée de l'IRI.

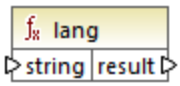
6.6.24 xslt | xpath functions

Les fonctions dans ce sous-groupe sont des fonctions XPath 1.0 qui extraient des informations concernant des items de mappage (ou des nœuds). La plupart de ces fonctions prennent un nœud en tant qu'argument et retournent des informations concernant ce nœud. Les fonctions **`last`** et **`position`** fonctionnent dans le contexte de mappage[407] actuel qui est déterminé par les connexions dans votre mappage.

Note : vous trouverez des fonctions XPath 1.0 supplémentaires dans la bibliothèque **core**.

6.6.24.1 lang

Retourne **true** si le nœud contextuel a un attribut `xml:lang` avec une valeur soit qui correspond exactement à l'argument **string**, ou en est un sous-ensemble. Sinon, la fonction retourne **false**.



Langages

XSLT 1.0.

Paramètres

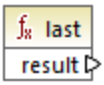
Nom	Type	echo Hello, World!
Paramètres	`xs:string`	Le code de langage à vérifier, par exemple, "en".

Exemple

Voir l'exemple donné pour la fonction **lang** 342 de la bibliothèque **xpath2**.

6.6.24.2 last

Retourne le numéro de la position du dernier nœud dans la liste de nœud traitée.



Langages

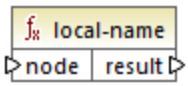
XSLT 1.0.

Exemple

Voir l'exemple donné pour la fonction **last** 324 de la bibliothèque **xpath2**.

6.6.24.3 local-name

Retourne la partie locale du nom du nœud fourni en tant qu'argument.



Langages

XSLT 1.0, XSLT 2.0, XSLT 3.0.

Paramètres

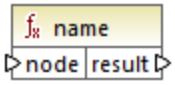
Nom	Type	Description
node	`node()`	Le nœud d'entrée.

Exemple

Voir l'exemple donné pour la fonction **local-name** 344 de la bibliothèque **xpath2**.

6.6.24.4 name

Retourne le nom du nœud fourni en tant qu'argument.



Langages

XSLT 1.0, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Type	Description
node	**node()**	Le nœud d'entrée.

Exemple

Voir l'exemple donné pour la fonction **local-name** 344 de la bibliothèque **xpath2**.

6.6.24.5 namespace-uri

Retourne l'espace de nom URI du nœud fourni en tant qu'argument.

Langages

XSLT 1.0, XSLT 2.0, XSLT 3.0.

Paramètres

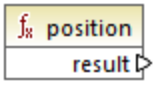
Nom	Type	Description
node	**node()**	Le nœud d'entrée.

Exemple

Voir l'exemple donné pour la fonction `local-name`[344] de la bibliothèque **xpath2**.

6.6.24.6 position

Retourne la position du nœud actuel dans le nœud qui est traité actuellement.



Langages

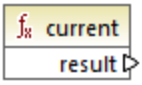
XSLT 1.0.

6.6.25 xslt | xslt functions

Les fonctions dans ce groupe sont des fonctions XSLT 1.0 diverses.

6.6.25.1 current

La fonction `current` ne prend aucun argument et retourne le nœud actuel.

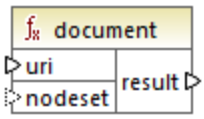


Langages

XSLT 1.0.

6.6.25.2 document

Accède aux nœuds depuis un document XML externe. Le résultat est sorti dans un nœud dans le document de sortie.



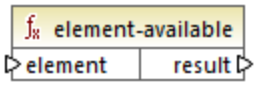
Langages
XSLT 1.0.

Paramètres

Nom	Type	echo Hello, World!
uri	`xs:string`	Obligatoire. Spécifie le chemin vers le document XML. Le document XML doit être valide et parsable.
nodeset	`node()`	Optionnel Spécifie un nœud, l'URI de base qui est utilisé pour résoudre l'URI fournie en tant que le premier argument s'il est relatif.

6.6.25.3 element-available

La fonction **`element-available`** teste si un élément, saisi en tant que le seul argument de string de la fonction, est pris en charge par le processeur XSLT. Le string d'argument est évalué en tant qu'un QName. C'est pourquoi, les éléments XSLT doivent avoir un préfixe `xsl:` et les éléments de Schéma XML doivent avoir un préfixe `xs:` —puisqu'ils s'agit des préfixes déclarés pour ces espaces de noms dans le XSLT sous-jacent qui sera généré pour le mappage.. La fonction retourne une booléenne.



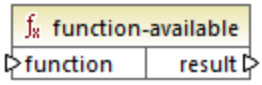
Langages
XSLT 1.0.

Paramètres

Nom	Type	echo Hello, World!
element	`xs:string`	Le nom d'élément.

6.6.25.4 function-available

La fonction **`function-available`** est semblable à la fonction **`element-available`** et teste si le nom de fonction est fourni en tant que l'argument de la fonction est pris en charge par le processeur XSLT. Le string d'entrée est évalué en tant qu'un QName. La fonction retourne une booléenne.



Langages

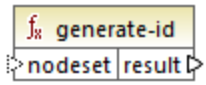
XSLT 1.0.

Paramètres

Nom	Type	echo Hello, World!
fonction	`xs:string`	Le nom de la fonction.

6.6.25.5 generate-id

La fonction **`generate-id`** génère un string unique qui identifie le premier nœud dans le nodeset identifié par l'argument d'entrée optionnel. Si aucun argument n'est fourni, l'ID est généré dans le nœud contextuel. Le résultat peut être dirigé à tout nœud dans le document de sortie.



Langages

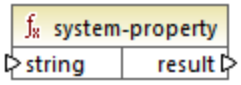
XSLT 1.0, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Type	Description
nodeset	`node()`	Argument optionnel qui fournit le nœud d'entrée.

6.6.25.6 system-property

La fonction **`system-property`** retourne des propriétés du processeur XSLT (le système). Trois propriétés de système, toutes dans l'espace de noms XSLT, sont obligatoires pour les processeurs XSLT. Il s'agit de `xsl:version`, `xsl:vendor` et `xsl:vendor-url`. Le string d'entrée est évalué en tant que QName et doit donc avoir le préfixe `xsl:` puisque le préfixe associé avec l'espace de noms XSLT dans la feuille de style XSLT sous-jacente.



Langages

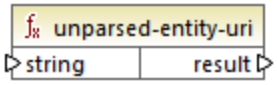
XSLT 1.0, XSLT 2.0, XSLT 3.0.

Paramètres

Nom	Type	Description
string	`xs:string`	Specifie le nom de propriété qui peut être un des suivants : `xsl:version`, `xsl:vendor`, `xsl:vendor-url`.

6.6.25.7 unparsed-entity-uri

Si vous utilisez un DTD, vous pouvez y déclarer une entité non parsée. Cette entité non parsée (par exemple une image) aura une URI qui situe l'entité non parsée. Le string d'entrée de la fonction doit correspondre au nom de l'entité non parsée qui a été déclarée dans le DTD. La fonction retourne ensuite l'URI de l'entité non parsée, qui peut ensuite être dirigée dans un nœud dans le document de sortie, par exemple dans un nœud **href**.



Langages

XSLT 1.0.

Paramètres

Nom	Type	echo Hello, World!
Paramètres	`xs:string`	Le nom de l'entité non parsée dont l'URI doit être extraite.

7 Scénarios de mappage avancé

Site web d'Altova : Outil d'intégration des données

Cette section décrit les scénarios de mappage avancé et inclut les rubriques suivantes :

7.1 Mapper noms de nœud

La plupart du temps, lorsque vous créez un mappage avec MapForce, l'objectif est de lire des *valeurs* depuis une source et d'écrire des *valeurs* dans une cible. Néanmoins, il peut y avoir des cas dans lesquels vous souhaitez accéder non seulement aux *valeurs* de nœud depuis la source, mais aussi aux noms de nœud. Par exemple, vous pouvez souhaiter créer un mappage qui écrit les noms d'élément ou d'attributs (pas des valeurs) depuis un XML de source et les convertit en des valeurs d'élément ou d'attribut (pas de noms) dans un XML cible.

Prenons l'exemple suivant : vous avez un fichier XML qui contient une liste de produits. Chaque produit possède le format suivant :

```
<product>
   <id>1</id>
   <color>red</color>
   <size>10</size>
</product>
```

Votre objectif est de convertir l'information concernant chaque produit dans des paires nom-valeur, par exemple :

```
<product>
   <attribute name="id" value="1" />
   <attribute name="color" value="red" />
   <attribute name="size" value="10" />
</product>
```

Pour ce type de scénario, vous souhaitez accéder au nom de nœud depuis le mappage. Avec l'accès *dynamique* aux noms de nœud, vous pouvez effectuer des conversions de données comme celle présentées ci-dessus.

Note : Vous pouvez aussi effectuer la transformation ci-dessus en utilisant les fonctions de bibliothèque principale **node-name**[272] et **static-node-name**[273]. Néanmoins, dans ce cas, vous devrez savoir exactement quels noms d'élément vous attendez de la source, et vous devrez connecter chaque élément manuellement vers la cible. Néanmoins, ces fonctions peuvent ne pas être suffisantes, par exemple si vous souhaitez filtrer ou regrouper des nœuds par leur nom, ou si vous souhaitez manipuler le type de données du nœud depuis le mappage.

L'accès à des noms de nœud dynamiquement est possible non seulement lorsque vous souhaitez lire des noms de nœud, mais aussi lorsque vous souhaitez les écrire. Dans un mappage standard, le nom des attributs ou des éléments dans une cible est toujours connu avant que le mappage soit exécuté ; il provient du schéma sous-jacent du composant. Avec des noms de nœud dynamiques, toutefois, vous pouvez créer de nouveaux attributs ou éléments dont le nom n'est pas connu avant l'exécution du mappage. En particulier, le nom de l'attribut ou de l'élément est fourni par le mappage lui-même, depuis toute source prise en charge par MapForce.

Pour que l'accès dynamique aux éléments ou attributs enfants d'un nœud soit possible, le nœud doit effectivement avoir des éléments ou des attributs enfant et ce ne doit pas être le nœud racine XML.

Les noms de nœud dynamiques sont pris en charge lorsque vous mappez depuis ou vers les types de composant suivants :

- XML
- CSV/FLF*

* Nécessite l'édition MapForce Professional ou Enterprise.

Les noms de nœud dynamiques sont pris en charge dans un des langages de mappage suivants : Built-In*, XSLT2,XSLT3 , XQuery*, C#*, C++*, Java*.

* Ces langages nécessitent l'édition MapForce Professional ou Enterprise.

Pour plus d'information concernant le fonctionnement des noms de nœud dynamiques, voir Obtenir l'accès aux noms de nœud 384. Pour un exemple de mappage étape par étape, voir Exemple : Mapper des noms d'élément vers les valeur d'attribut 396.

7.1.1 Obtenir l'accès aux noms de nœud

Lorsqu'un nœud dans un composant XML a des nœuds enfants, vous pouvez obtenir le nom et la valeur de chaque nœud enfant directement dans le mappage. Cette technique est appelée "noms de nœud dynamiques". "Dynamique" se réfère au fait que le traitement a lieu "on the fly" (immédiatement), pendant l'exécution du mappage et n'est pas basé sur l'information de schéma statique qui est connu avant l'exécution du mappage. Cette rubrique présente des détails pour permettre l'accès dynamique aux noms de nœud et montre ce que vous pouvez en faire.

Lorsque vous lisez des données provenant d'une source, "noms de nœud dynamiques" signifie que vous pouvez faire les opérations suivantes :

- Obtenir une liste de tous les nœuds enfants (ou attributs) d'un nœud, en tant que séquence. Dans MapForce, "séquence" est une liste de zéro items ou plus que vous pouvez connecter à une cible et créer autant d'items dans la cible qu'il y a d'items dans la source. Ainsi, par exemple, si un nœud a cinq attributs dans la source, vous pouvez créer cinq nouveaux éléments dans la cible, chacun correspondant à un attribut.
- Lire non seulement les valeurs de nœud enfants (comme un mappage standard le fait), mais aussi leurs noms.

Lorsque vous écrivez des données dans une cible, "noms de nœud dynamiques" signifie que vous pouvez effectuer les opérations suivantes :

- Créer de nouveaux nœuds en utilisant des noms fournis par le mappage (des noms soit-disant "dynamiques"), par opposition aux noms fournis par les paramètres de composant (des noms soit-disant "static").

Afin d'illustrer les noms de nœud dynamiques, cette rubrique utilise le schéma XML suivant : **<Documents>\Altova\MapForce2024\MapForceExamples\Tutorial\Products.xsd**. Ce schéma est accompagné par un document d'instance échantillon, **Products.xml**. Pour ajouter le schéma et le fichier d'instance à la zone de mappage, choisir la commande de menu **Insérer | Schéma XML /Fichier** et chercher **<Documents>\Altova\MapForce2024\MapForceExamples\Tutorial\Products.xml**. Lorsque vous serez invité à choisir un élément racine, choisir `products`.

Pour activer les noms de nœud dynamiques pour le nœud `product`, cliquer dessus avec la touche de droite et choisir parmi une des commandes du menu contextuel suivantes :

- **Afficher les attributs avec un Nom Dynamique**, si vous souhaitez obtenir l'accès aux attributs du nœud
- **Afficher les éléments enfants avec un Nom Dynamique**, si vous souhaitez obtenir l'accès aux éléments enfants du nœud

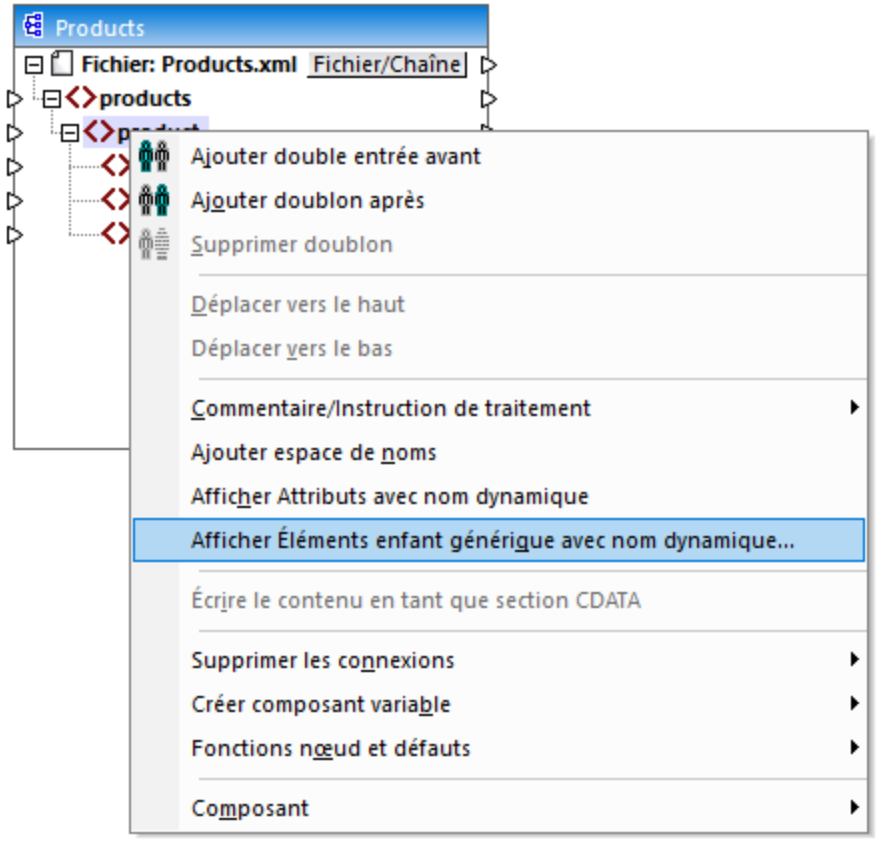


Fig. 1 Activer des noms de nœud dynamiques (pour les éléments enfant)

Note : Les commandes ci-dessus sont disponibles uniquement pour les nœuds qui ont des nœuds enfants. De même, les commandes ne sont pas disponibles pour les nœuds de racine.

Lorsque vous faites passer un nœud dans le mode dynamique, un dialogue comme celui ci-dessous apparaît. Définir les options comme indiqué ci-dessous ; ces options seront présentées plus en détail dans Accéder les nœuds de type spécifique 392.

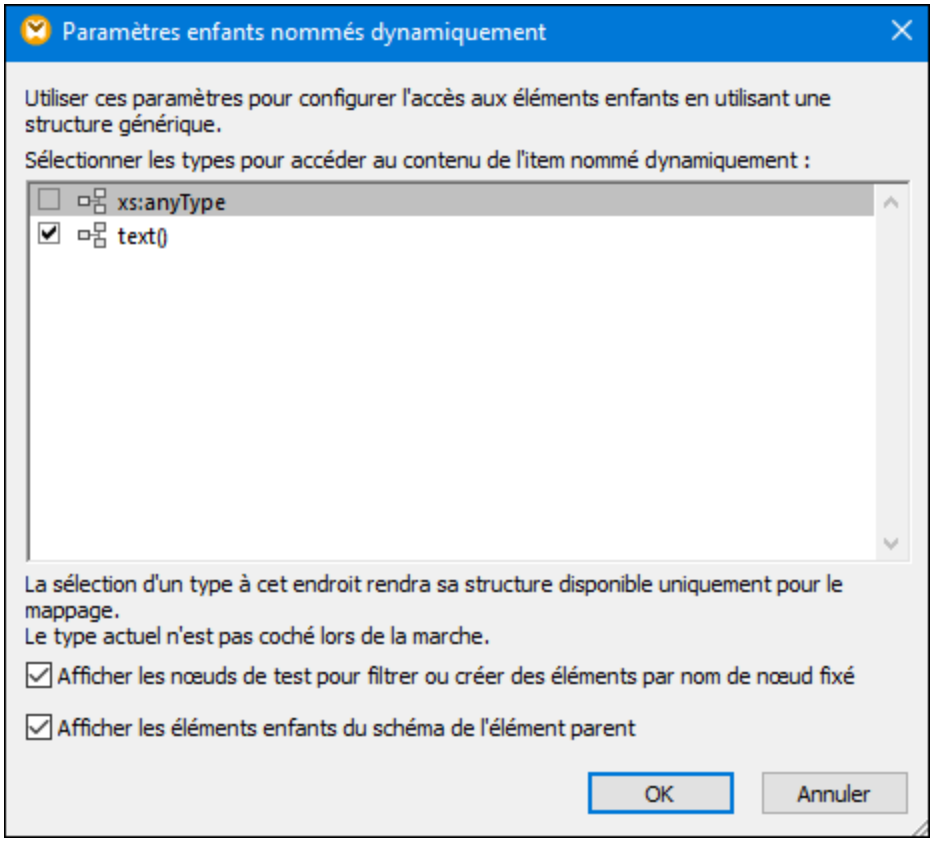


Fig. 2 Dialogue "Paramètres d'Enfants nommés dynamiquement"

Fig. 3 Montre à quoi ressemble le composant lorsque les noms de nœud dynamiques sont activés pour le nœud `product`. Veuillez noter que l'apparence du composant a considérablement changé.

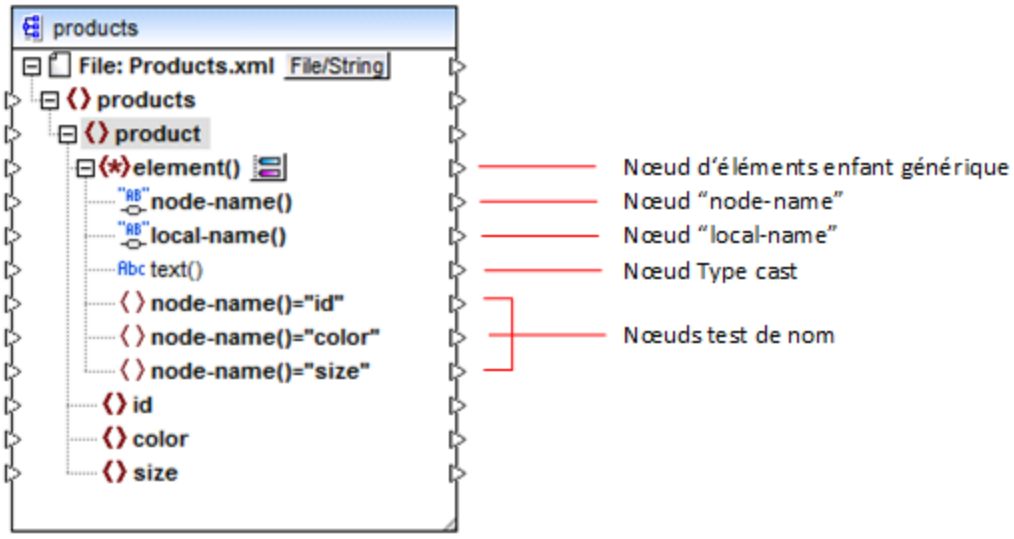


Fig.3 Noms de nœud dynamiques activés (pour les éléments)

Pour faire retourner le composant dans le mode standard, cliquer avec la touche de droite sur le nœud `product`, et désactiver l'option **Afficher les Éléments enfants avec le nom dynamique** depuis le menu contextuel.

L'image ci-dessous montre à quoi ressemble le même composant lorsque l'accès dynamique aux attributs d'un nœud est activé. Le composant a été obtenu en cliquant avec la touche de droite sur l'élément `product`, et en sélectionnant **Afficher les attributs avec un Nom dynamique** depuis le menu contextuel.

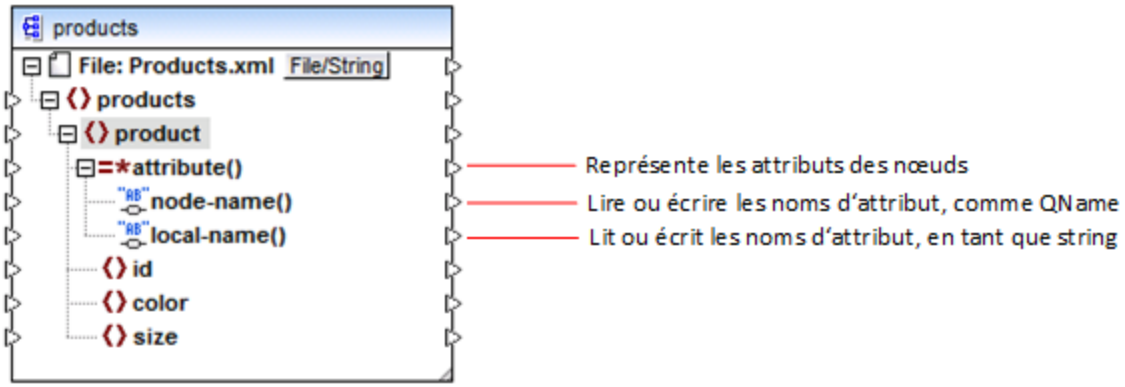


Fig. 4 Noms de nœud dynamiques activés (pour les attributs)

Pour replacer le composant dans le mode standard, cliquer avec la touche de droite dans le mode `product`, et désactiver l'option **Afficher les attributs avec un Nom dynamique** depuis le menu contextuel.

Comme illustré dans les Fig. 3 et Fig. 4, l'apparence du composant change lorsqu'un nœud (dans ce cas, `product`) est placé en mode "nom de nœud dynamique". La nouvelle apparence ouvre des possibilités pour les actions suivantes :

- Lire ou écrire une liste de tous les éléments ou attributs enfants d'un nœud. Ils sont fournis par l'item `element()` ou `attribute()`, respectivement.
- Lire ou écrire le nom de chaque élément ou attribut enfants. Le nom est fourni par les items `node-name()` et `local-name()`.
- Dans le cas des éléments, lire ou écrire la valeur de chaque élément enfant, en tant que type de données spécifique. Cette valeur est fournie par le nœud de type cast (dans ce cas, l'item `text()`). Veuillez noter que seuls des éléments peuvent avoir des nœuds type cast. Les attributs sont toujours traités en tant que type "string".
- Regrouper ou filtrer les éléments enfants par nom.

Les types de nœud avec lesquels vous pouvez travailler dans le mode "nom de nœud dynamique" sont décrits ci-dessous.

element()

Ce nœud a un comportement différent dans un composant de source comparé à un composant cible. Dans un composant de source, il fournit les éléments enfant du nœud en tant que séquence. Dans la Fig.3, `element()` fournit une liste (séquence) de tous les éléments enfants de `product`. Par exemple, la séquence créée depuis le XML suivant contiendra trois items (puisqu'il y a trois éléments enfant de `product`) :

```
<product>
   <id>1</id>
   <color>red</color>
   <size>10</size>
</product>
```

Veuillez noter que le nom et le type de chaque item dans la séquence est fourni par le nœud `node-name()` et le nœud type cast, respectivement (discuté ci-dessous). Pour comprendre cela, imaginez que vous souhaitez transformer des données depuis un XML de source dans un XML cible comme suit :

Fig. 6 Mapper des noms d'élément XML dans les valeurs d'attribut (exigence)

Le mappage qui atteindrait cet objectif comme suit :

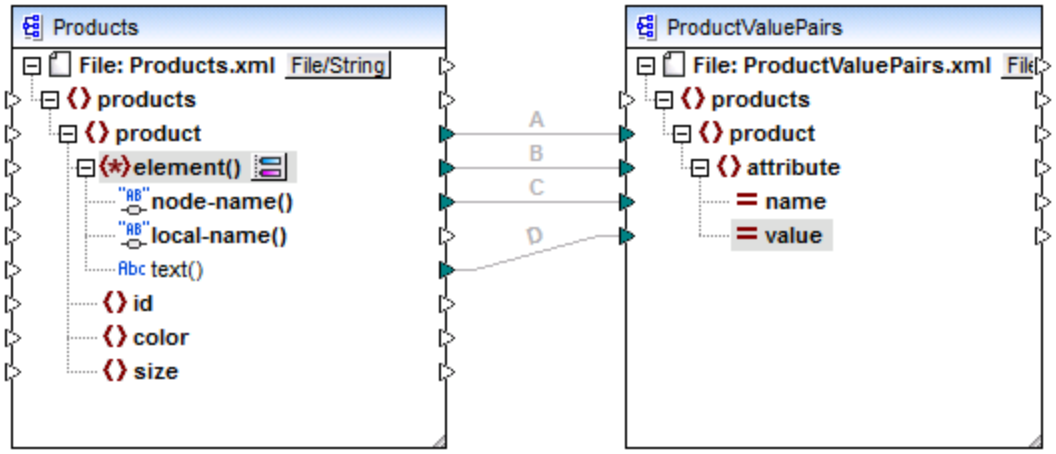


Fig. 7 Mapper des noms d'élément XML dans les valeurs d'attribut (dans MapForce)

Le rôle d'`element()` ici est de fournir la séquence des éléments enfants de `product`, alors que `node-name()` et `text()` fournissent le nom et la valeur de chaque item dans la séquence. Ce mappage est accompagné par un échantillon de tutoriel et est discuté plus en détail dans Exemple : Mapper noms d'élément dans les valeurs d'attributs 396.

Dans un composant cible, `element()` ne crée rien par lui-même, ce qui est une exception à la règle de base du mappage "pour chaque item dans la source, créer un item cible". Les éléments sont créés par les nœuds type cast (en utilisant la valeur de `node-name()`) et par les nœuds de test de nom (utilisant leur propre nom).

attribute()

Comme indiqué dans la Fig. 4, cet item permet l'accès à tous les attributs du nœud, à l'exécution du mappage. Dans un composant de source, il fournit les attributs du nœud de source connecté, en tant que séquence. Par exemple, dans le XML suivant, la séquence contiendrait deux items (puisque `product` a deux attributs) :

```
<product id="1" color="red" />
```

Veuillez noter que le nœud `attribute()` fournit uniquement la valeur de chaque attribut dans la séquence, toujours en tant que type string. Le nom de chaque attribut est fourni par le nœud `node-name()`.

Dans un composant cible, ce nœud traite une séquence connectée et crée une valeur d'attribut pour chaque item dans la séquence. Le nom d'attribut est fourni par `node-name()`. Par exemple, imaginer que vous souhaitez transformer des données depuis un XML de source dans un XML cible comme suit :

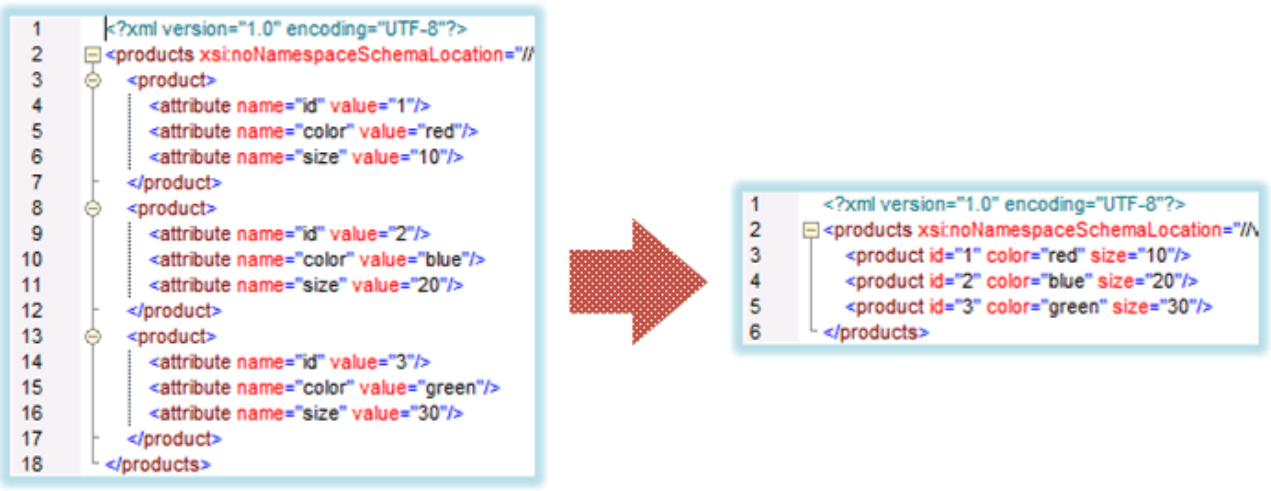


Fig. 8 Mapper des valeurs d'attribut pour attribuer des noms (exigence)

Le mappage qui permet d'atteindre cet objectif ressemble à ce qui suit :

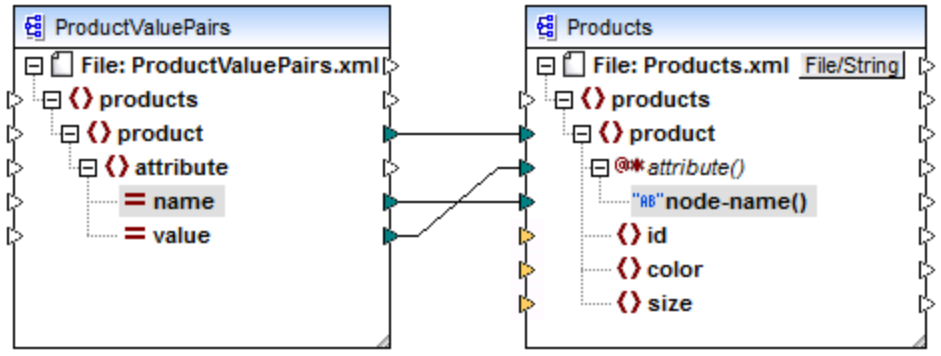


Fig. 9 Mapper des valeurs d'attribut pour attribuer des noms (dans MapForce)

Note : Cette transformation est également possible sans activer un accès dynamique dans des attributs d'un nœud. Ici, elle illustre comment `attribute()` fonctionne dans un composant cible.

Si vous souhaitez reconstruire ce mappage, il utilise les mêmes composants XML que le mappage **ConvertProducts.mfd** disponible dans le dossier **<Documents>\Altova\MapForce2024\MapForceExamples\Tutorial**. La seule différence est que la cible est maintenant devenue la source, et que la source est devenue la cible. En guise de données d'entrée pour le composant source, vous nécessiterez une instance XML qui contient effectivement des valeurs d'attribut, par exemple :

```
<?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>
```

Veuillez noter que, dans l'extrait de code, l'espace de noms et la déclaration de schéma ont été omis, pour des raisons de simplicité.

node-name()

Dans un composant source, `node-name()` fournit le nom de chaque élément enfant de `element()`, ou le nom de chaque attribut de `attribute()`, respectivement. Par défaut, le nom fourni est de type `xs:QName`. Pour obtenir le nom en tant que string, utiliser le nœud `local-name()` (voir Fig. 3).

Dans un composant cible, `node-name()` écrit le nom de chaque élément ou attribut contenu dans `element()` ou `attribute()`.

local-name()

Ce nœud fonctionne de la même manière que `node-name()`, à la différence que le type est `xs:string` au lieu de `xs:QName`.

Nœud type cast

Dans un composant source, le nœud type cast fournit la valeur de chaque élément enfant contenu dans `element()`. Le nom et la structure de ce nœud dépendent du type sélectionné depuis le dialogue "Paramètres Enfants nommés dynamiquement" (Fig. 2).

Pour changer le type du nœud, cliquer sur la touche **Changer sélection** ( ) et choisir un type depuis la liste des types disponibles, y compris un caractère générique de schéma (`xs:any`). Pour plus d'informations, voir Accéder aux nœuds de type spécifique 392.

Dans un composant cible, le nœud type cast écrit la valeur de chaque élément enfant contenu dans `element()`, en tant que type de données spécifique. De même, le type de données désiré peut être sélectionné en cliquant sur la touche **Changer sélection** ( ).

Nœuds de test de nom

Dans un composant de source, les nœuds de test de nom proposent un moyen pour regrouper ou filtrer les éléments enfants depuis une instance de source par nom. Vous pouvez souhaitez filtrer les éléments enfants par les noms pour vous assurer que le mappage accède aux données d'instance en utilisant le type correct (voir Accéder aux nœuds de type spécifique 392).

En général, les nœuds de test de nom fonctionnent presque comme des nœuds d'élément normaux pour lire et écrire des valeurs et des structures sous-arborescentes. Néanmoins, étant donné que la sémantique de mappage est différente lorsque l'accès dynamique est activé, il existe des limites. Par exemple, vous ne pouvez pas concaténer la valeur de deux nœuds de test de nom.

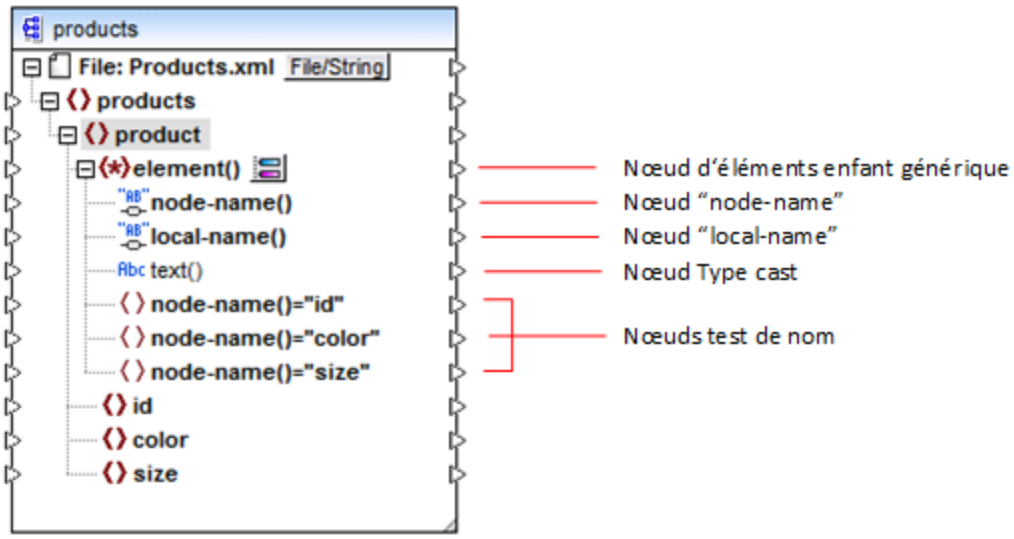
Du côté cible, les nœuds de test de nom créent autant d'éléments dans la sortie qu'il y a d'items dans la séquence de source connectée. Leur nom contourne la valeur mappée dans `node-name()`.

Si nécessaire, vous pouvez dissimuler les nœuds de test de nom depuis le composant. Pour ce faire, cliquer sur la touche **Changer la sélection** ( ) à côté du nœud `element()`. Ensuite, dans le dialogue

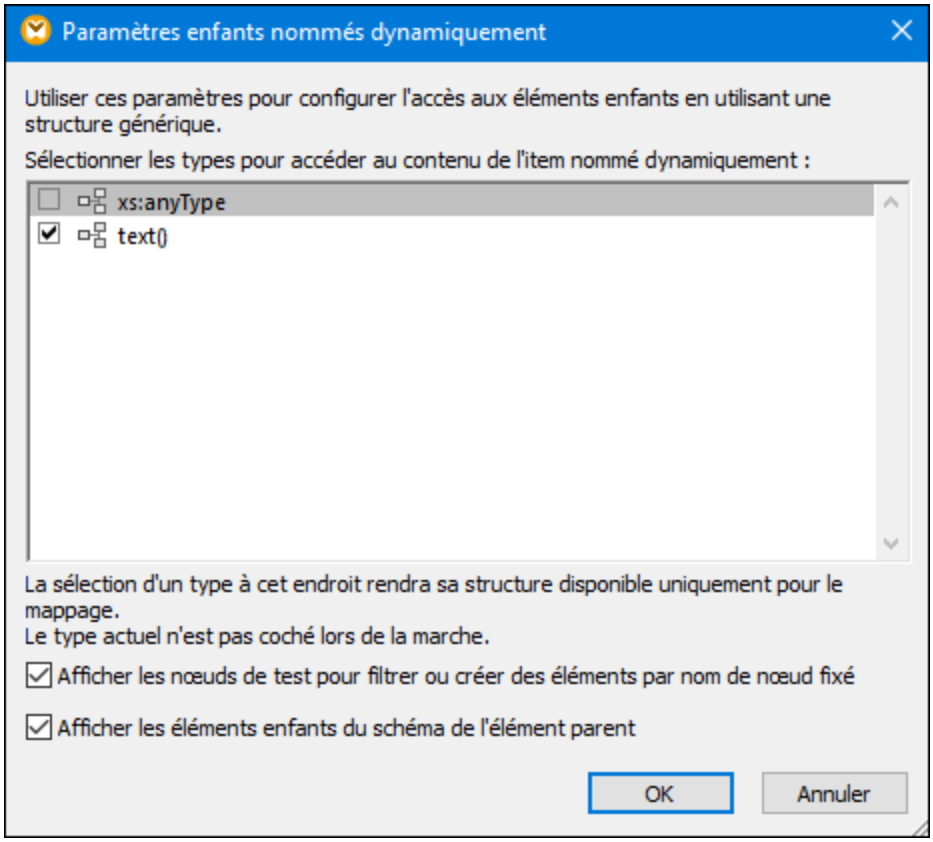
"*Paramètres Enfants nommés dynamiquement*" (Fig. 2), décocher la case **Afficher les nœuds de test de nom...**.

7.1.2 Obtenir l'accès aux nœuds de type spécifique

Comme mentionné dans la section précédente, Obtenir l'accès aux noms de nœud 384, vous pouvez accéder à tous les éléments enfants d'un nœud en cliquant avec la touche de droite et sélectionner la commande de menu contextuel **Afficher les éléments enfants avec un Nom Dynamique**. Au moment de l'exécution de mappage, cela entraîne une accessibilité du nom de chaque élément enfant par le biais du nœud `node-name()`, alors que la valeur est accessible par le biais d'un nœud type cast spécial. Dans l'image ci-dessous, le nœud type cast est le nœud `text()`.



Attention, le type de données de chaque élément enfant n'est pas connu avant l'exécution de mappage. Du reste, il peut être différent pour chaque élément enfant. Par exemple, un nœud `product` dans le fichier d'instance XML peut avoir un élément enfant `id` de type `xs:integer` et un élément enfant `size` de type `xs:string`. Afin de vous permettre d'accéder au contenu de nœud d'un type spécifique, le dialogue affiché ci-dessous s'ouvre à chaque fois que vous activez l'accès dynamique dans des éléments enfant d'un nœud. Vous pouvez aussi ouvrir ce dialogue à tout moment ultérieurement, en cliquant sur la touche **Changer sélection** ( ) située à côté du nœud `element()`.



Dialogue "Paramètres d'Enfants nommés dynamiquement"

Pour accéder au contenu de chaque élément enfant au moment d'exécution de mappage, vous disposez de plusieurs options :

1. Accéder au contenu en tant que string. Pour ce faire, sélectionner la case à cocher **text()** dans le dialogue ci-dessus. Dans ce cas, un nœud `text()` est créé dans le composant lorsque vous refermez le dialogue. Cette option est appropriée si le contenu est de type simple (`xs:int`, `xs:string`, etc.) et est illustré dans l'Exemple : Mapper les noms d'élément dans les valeurs d'attribut 396. Veuillez noter qu'un nœud **text()** est uniquement affiché si un nœud enfant du nœud actuel peut contenir du texte.
2. Accéder au contenu en tant qu'un type complexe particulier autorisé par le schéma. Lorsque des types complexes personnalisés définis globalement sont autorisés par le schéma pour le nœud sélectionné, ils sont aussi disponibles dans le dialogue ci-dessus, et vous pouvez cocher la case adjacente. Dans l'image ci-dessus, il n'y a pas de types complexes définis globalement par le schéma, pour qu'aucun ne soit disponible pour la sélection.
3. Accéder au contenu en tant que type any. Cela peut être utile dans des scénarios dans le mappage avancé (voir "Accéder aux structures profondes" ci-dessous). Pour ce faire, sélectionner la case à cocher située à côté de **xs:anyType**.

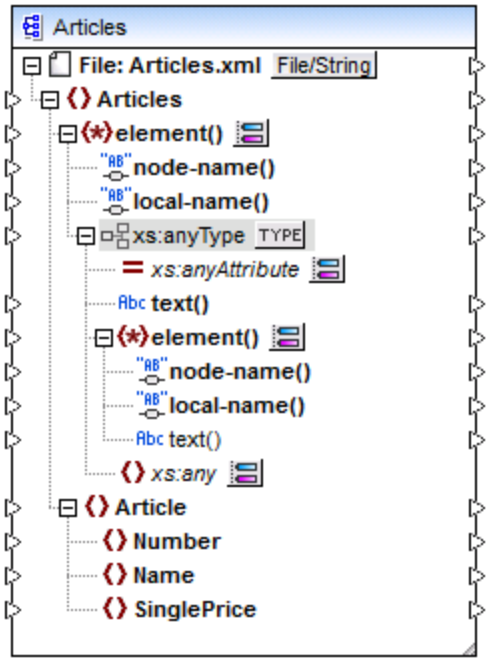
Veuillez noter que lors de l'exécution de mappage, MapForce (par le biais du nœud type cast) n'a pas d'informations par rapport à ce qu'est le type réel du nœud d'instance. C'est pourquoi, votre mappage doit accéder au contenu de nœud en utilisant le type correct. Par exemple, si vous prévoyez que le nœud

d'une instance XML de source peut avoir des nœuds enfants de plusieurs types complexes, procéder comme suit :
a) Définir le nœud type cast pour être le type complexe que vous souhaitez faire correspondre (voir item 2 dans la liste ci-dessus).
b) Ajouter un filtre à lire depuis l'instance uniquement par le type complexe que vous souhaitez faire correspondre. Pour plus d'informations concernant les filtres, voir Filtres et Conditions 176.

Accéder aux structures plus profondes

Il est possible d'accéder à des nœuds à des niveaux plus profonds dans le schéma qu'au niveau des enfants immédiats d'un nœud. Cela est utile dans les scénarios de mappage avancés. Dans des mappages simples comme Exemple : Mapper les noms d'élément sur des valeurs d'attribut 396, vous n'aurez pas besoin de cette technique car le mappage accède uniquement aux enfants immédiats d'un nœud XML. Néanmoins, si vous souhaitez accéder dynamiquement aux structures plus profondes comme les "petits-enfants", les "arrière-petits-enfants", etc. suivez les instructions ci-dessous :

1. Créer un nouveau mappage.
2. Dans le menu Insérer, cliquer sur **Insérer Schéma XML/Fichier** et chercher le fichier d'instance XML (dans cet exemple, le fichier **Articles.xml** provenant du dossier **<Documents>\Altova\MapForce2024\MapForceExamples\Tutorial**).
3. Cliquer avec la touche de droite sur le nœud `Articles` et choisir la commande contextuelle **Afficher les éléments enfant avec un nom dynamique**.
4. Choisir **xs:anyType** depuis le dialogue "Paramètres Enfants nommés dynamiquement".
5. Cliquer avec la touche de droite sur le nœud `xs:anyType` et choisir à nouveau la commande contextuelle **Afficher les éléments enfant avec un nom dynamique**.
6. Choisir **text()** depuis le dialogue "Paramètres Enfants nommés dynamiquement".



Dans le composant ci-dessus, veuillez noter qu'il y a deux nœuds `element()`. Le second nœud `element()` propose un accès dynamique aux petits-enfants de l'instance `<Articles>` **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

Par exemple, pour obtenir les noms d'élément "petits-enfants" (`Number`, `Name`, `SinglePrice`), vous pouvez établir une connexion depuis le nœud `local-name()`sous le deuxième nœud `element()` vers un item cible. De même, pour obtenir des valeurs d'élément "petits-enfants" (`1`, `T-Shirt`, `25`), vous pouvez établir une connexion depuis le nœud `text()`.

Bien que cela ne s'applique pas à cet exemple, dans des situations réelles, vous pouvez aussi activer des noms de nœud dynamiques pour tout nœud `xs:anyType` ultérieur, pour atteindre des niveaux plus profonds.

Veuillez noter les points suivants :

- La touche TYPE vous permet de choisir tout type dérivé depuis le schéma actuel et de l'afficher dans un nœud séparé. Cela peut uniquement être utile si vous souhaitez mapper vers ou depuis des types de schéma dérivés (voir Types de Schéma XML dérivé 118).
- La touche **Changer sélection** ( ) située à côté d'un nœud `element()` ouvre le dialogue "Paramètres Enfants nommés dynamiquement" discuté dans cette rubrique.
- La touche **Changer sélection** ( ) située à côté de `xs:anyAttribute` vous permet de choisir tout attribut défini globalement dans le schéma. De même, la touche **Changer sélection** ( ) située à côté de `xs:any` permet de choisir tout élément défini globalement dans le schéma. Cela fonctionne de la même manière que le fait de mapper depuis ou vers des caractères génériques de schéma (voir aussi Caractères génériques - xs:any / xs:anyAttribute 124). Si vous utilisez cette option, assurez-vous que l'attribut ou l'élément sélectionné peut réellement exister à ce niveau spécifique conformément au schéma.

7.1.3 Exemple : Mapper les noms d'élément dans les valeurs d'attribut

Cet exemple vous montre comment mapper des noms d'élément depuis un document XML pour attribuer des valeurs dans un document XML cible. L'exemple s'accompagne d'un mappage d'échantillon, qui est disponible dans le chemin suivant :
<Documents>\Altova\MapForce2024\MapForceExamples\Tutorial\ConvertProducts.mfd.

Afin de comprendre ce que fait l'exemple, partons du principe que vous avez un fichier XML qui contient une liste de produits. Chaque produit comporte le format suivant :

```
<product>
   <id>1</id>
   <color>red</color>
   <size>10</size>
</product>
```

Votre objectif est de convertir l'information concernant chaque produit en des paires nom-valeur, par exemple :

```
<product>
   <attribute name="id" value="1" />
   <attribute name="color" value="red" />
   <attribute name="size" value="10" />
</product>
```

Pour effectuer un mappage de données comme celui présenté ci-dessus avec un effort minimum, cet exemple utilise une fonction MapForce appelée "Accès dynamique aux noms de nœud". "Dynamique" signifie ici que, lorsque le mappage est exécuté, il peut lire les noms de nœud (pas uniquement les valeurs) et utilise ces noms en tant que valeurs. Vous pouvez créer le mappage requis en quelques étapes, voir ci-dessous.

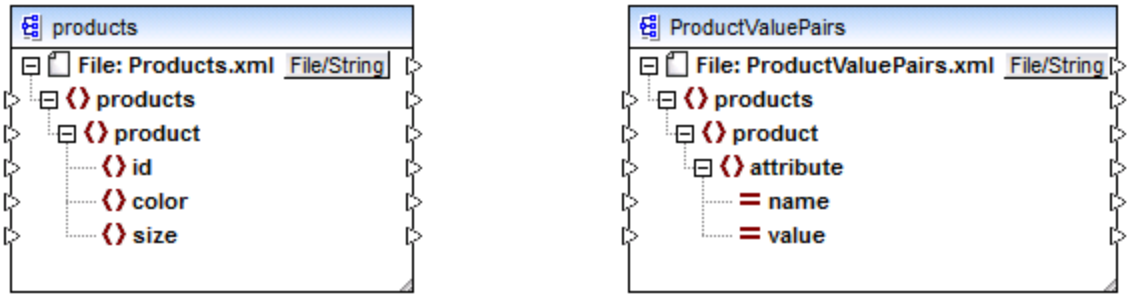
Étape 1: Ajouter le composant XML source au mappage

- Dans le menu **Insérer**, cliquer sur **Schéma XML/Fichier**, et chercher le fichier suivant : **<Documents>\Altova\MapForce2024\MapForceExamples\Tutorial\Products.xml**. Ce fichier XML pointe vers le schéma **Products.xsd** situé dans le même dossier.

Étape 2: Ajouter le composant XML cible au mappage

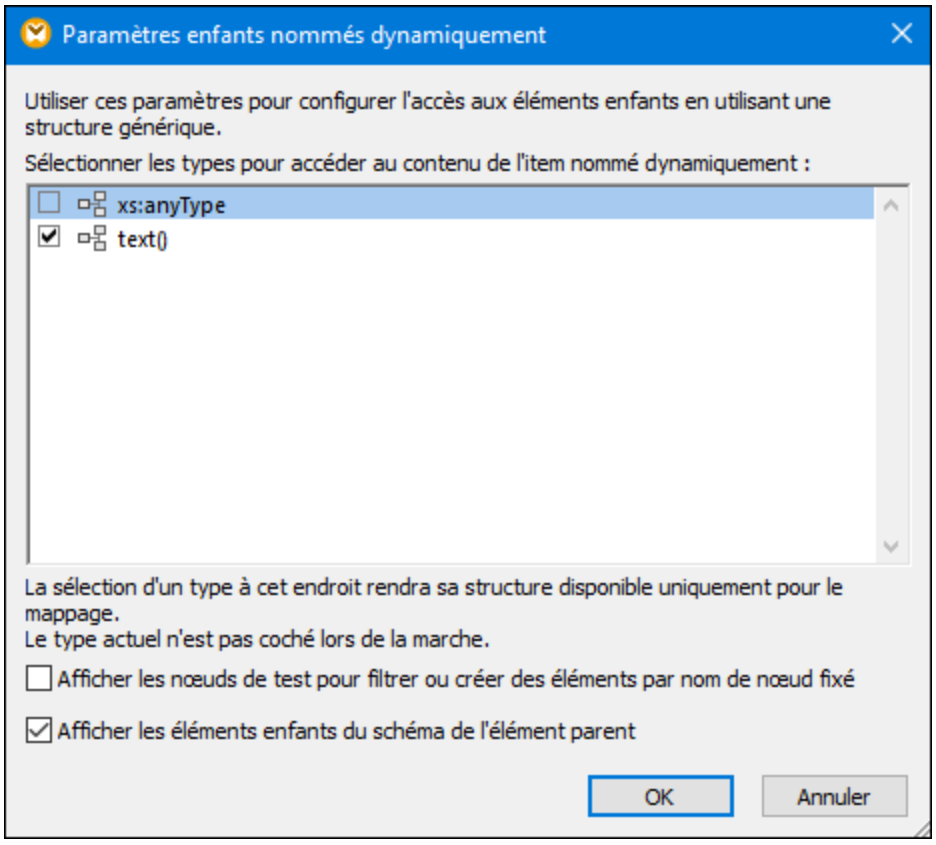
- Dans le menu **Insérer**, cliquer sur **Schéma XML/Fichier**, et chercher le fichier suivant : **<Documents>\Altova\MapForce2024\MapForceExamples\Tutorial\ProductValuePairs.xsd**. Lorsque vous êtes invité à fournir un fichier d'instance, cliquer sur **Sauter**. Lorsque vous êtes invité à choisir un élément racine, choisir `products` en tant qu'élément racine.

Á ce niveau, le mappage devrait ressembler à l'exemple suivant :

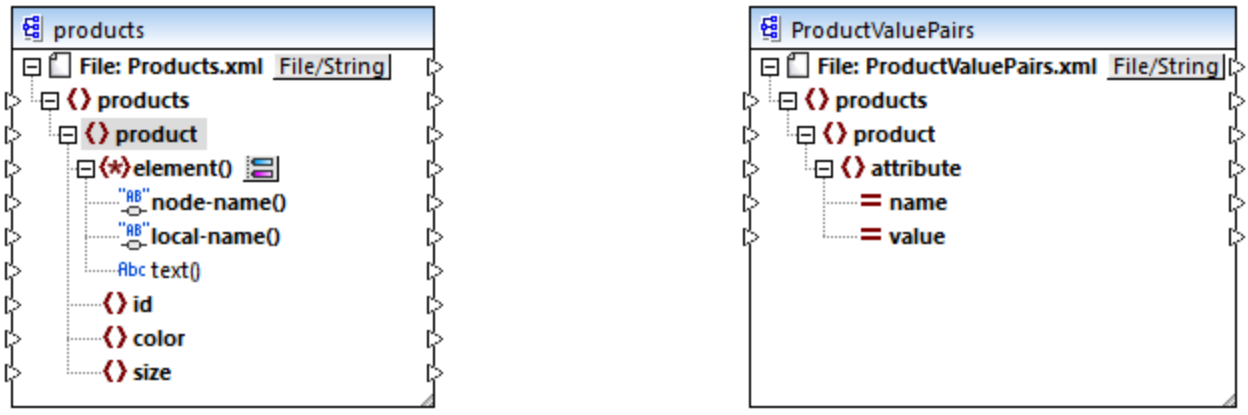


Étape 3: Activer l'accès dynamique aux nœuds enfants

1. Cliquer avec la touche de droite sur le nœud `product` dans le composant de source, et choisir **Afficher les éléments enfants avec un Nom Dynamique** depuis le menu contextuel.
2. Dans le dialogue qui s'ouvre, choisir **text()** en tant que type. Ne pas toucher aux autres options.

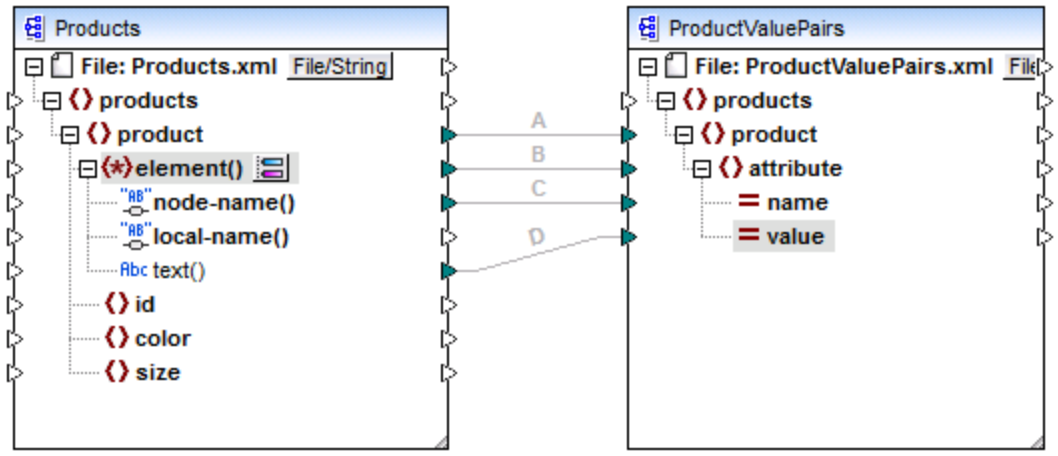


Veuillez noter qu'un nœud `text()` a été ajouté au composant de source. Ce nœud fournira le contenu de chaque item enfant au mappage (dans ce cas, la valeur de "id", "color", et "size").



Étape 4: Tracer les connexions de mappage

Enfin, tracer les connexions de mappage A, B, C, D comme illustré ci-dessous. En option, double-cliquer sur chaque connexion, en commençant par la connexion supérieure et en saisissant le texte "A", "B", "C", et "D", respectivement, dans le champ Description.



ConvertProducts.mfd

Dans le mappage illustré ci-dessus, la connexion A crée, pour chaque produit dans la source, un produit dans la cible. Jusqu'à présent, il s'agit d'une connexion MapForce standard qui n'adresse les noms de nœud d'aucune manière. La connexion B, néanmoins, crée pour chaque élément enfant de `product` rencontré, un nouvel élément dans la cible, appelé `attribute`.

La connexion B est une connexion cruciale dans le mappage. Nous rappelons ici l'objectif de cette connexion : elle porte une *séquence* des éléments enfants de `product` depuis la source vers la cible. Elle ne porte pas les *noms* ou les *valeurs*. Ainsi : si l'**element()** source a N éléments enfant, créer N instances de cet item dans la cible. Dans ce cas particulier, `product` dans la source a trois éléments enfant (`id`, `color` et `size`). Cela signifie que chaque `product` dans la cible aura trois éléments enfants portant le nom `attribute`.

Bien que non illustré dans cet exemple, la même règle est utilisée pour mapper les éléments enfant de **attribute()**: si l'item de source **attribute()** a N attributs enfants, créer N instances de cet item dans la cible.

Ensuite, la connexion C copie le nom de chaque élément enfant de `product` vers la cible (littéralement, "id", "color", et "size").

Enfin, la connexion D copie la valeur de chaque élément enfant du produit, en tant que type de string, dans la cible.

Pour consulter la sortie de mappage, cliquer sur l'onglet **Sortie** et observer le XML généré. Comme prévu, la sortie contient plusieurs produits dont les données sont stockées en tant que paires nom-valeur, ce qui était l'objectif de ce mappage.

```
<?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>
```

Sortie de mappage générée

7.2 Fichiers Batch-Process

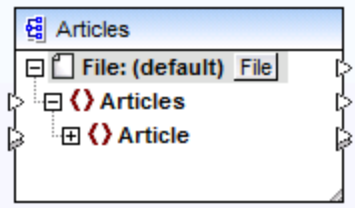
Vous pouvez configurer MapForce pour qu'il traite plusieurs fichiers (par exemple, tous les fichiers dans un répertoire) lorsque le mappage est en cours. L'utilisation de cette fonction vous permettra de résoudre les tâches suivantes :

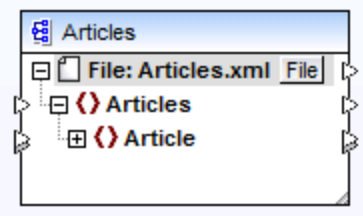
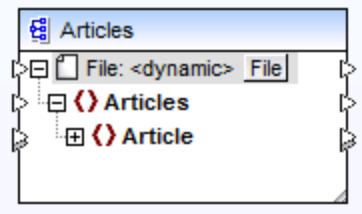
- Fournir dans le mappage une liste des fichiers d'entrée à traiter
- Générer en tant que sortie de mappage une liste des fichiers au lieu du fichier de sortie unique
- Générez une application de mappage où les noms de fichiers d'entrée et de sortie sont définis au moment de l'exécution
- Convertissez un ensemble de fichiers vers un autre format
- Partager un fichier volumineux dans des parties plus petites
- Fusionner plusieurs fichiers en un grand fichier

Vous pouvez configurer un composant MapForce pour traiter plusieurs fichiers d'une des manières suivantes :

- Fournir le chemin vers le(s) fichier(s) d'entrée ou de sortie en utilisant des caractères génériques au lieu d'un nom de fichier fixe, dans les paramètres de composant (voir Changer les paramètres du composants 39). Vous pouvez saisir les caractères génériques * et ? dans la boîte de dialogue des Paramètres du composant, pour que MapForce résout le chemin correspondant quand le mappage est exécuté.
- Connectez une séquence au nœud racine d'un composant qui forunit le chemin de manière dynamique (par exemple, le résultat de la fonction `replace-fileext`). Lorsque le mappage est exécuté, MapForce lira dynamiquement tous les fichiers d'entrée ou générer dynamiquement tous les fichiers de sortie.

Selon les objectifs que vous souhaitez atteindre, vous pouvez utiliser soit une de ces approches ou les deux sur le même mappage. Néanmoins, il n'est pas efficace d'utiliser les deux approches en même temps sur le même composant. Pour indiquer à MapForce quelle approche vous souhaitez utiliser pour un composant particulier, cliquer sur la touche **Fichier** (File) ou **Fichier/String** (File/String) disponible à côté du nœud racine d'un composant. Cette touche vous permet de spécifier le comportement suivant :

Utiliser les noms de fichier depuis les Paramètres de composant	Si le composant doit traiter un ou plusieurs fichiers d'instance, cette option indique à MapForce de traiter le(s) nom(s) de fichier définis dans le dialogue Paramètres de composant. Si vous choisissez cette option, le nœud de racine ne nécessite pas de connecteur d'entrée, étant donné que cela n'est pas utile. 

	Si vous n'avez pas spécifié de fichiers d'entrée ou de sortie dans le dialogue Paramètres de composant, le nom du nœud racine est **Fichier : (défaut)**. Sinon, le nœud racine affiche le nom du fichier d'entrée, suivi par un point-virgule (;), suivi par le nom du fichier de sortie. Si le nom de l'entrée est le même qu'avec celui du fichier de sortie, il sera affiché en tant que nom du nœud racine.  Veuillez noter que vous pouvez choisir soit cette option ou l'option *Utiliser les noms de fichier dynamiques fournis par le mappage*.
Utiliser les noms de fichier dynamiques fournis par le mappage	Cette option indique à MapForce de traiter le(s) nom(s) de fichier que vous définissez dans la zone de mappage, en connectent les valeurs au nœud de racine du composant. Si vous choisissez cette option, le nœud de racine reçoit un connecteur d'entrée auquel vous pouvez connecter des valeurs qui fournissent dynamiquement les noms de fichier à traiter pendant l'exécution de mappage. Si vous avez aussi défini les noms de fichier dans le dialogue Paramètres de composant, ces valeurs sont ignorées. Lorsque cette option est sélectionnée, le nom du nœud de racine est affiché en tant que **Fichier : <dynamic>**.  Cette option est mutuellement exclusive avec l'option *Utiliser Noms de fichier depuis les Paramètres de composant*.

Plusieurs fichiers d'entrée ou de sortie peuvent être définis pour les composants suivants :

- Fichiers XML
- Fichiers texte (CSV*, fichiers FLF* et fichiers FlexText**)

- Documents EDI**
- Feuilles de calcul Excel**
- Documents XBRL**
- Fichiers JSON**
- Fichiers Protocol Buffers**

* *Nécessite MapForce Professional Edition*
** *Nécessite MapForce Enterprise Edition*

La table suivante illustre la prise en charge pour les fichiers d'entrée et de sortie dynamiques et les caractères génériques dans les langages MapForce.

Langage cible	**Nom de fichier d'entrée dynamique**	La **prise en charge du caractère générique pour le nom de fichier d'entrée**	**nom de fichier de sortie dynamique**
XSLT 1.0	*	N'est pas pris en charge par XSLT 1.0	N'est pas pris en charge par XSLT 1.0
XSLT 2.0	*	*(1)	*
XSLT 3.0	*	*(1)	*
C++	*	*	*
C#	*	*	*
Java	*	*	*
BUILT-IN	*	*	*

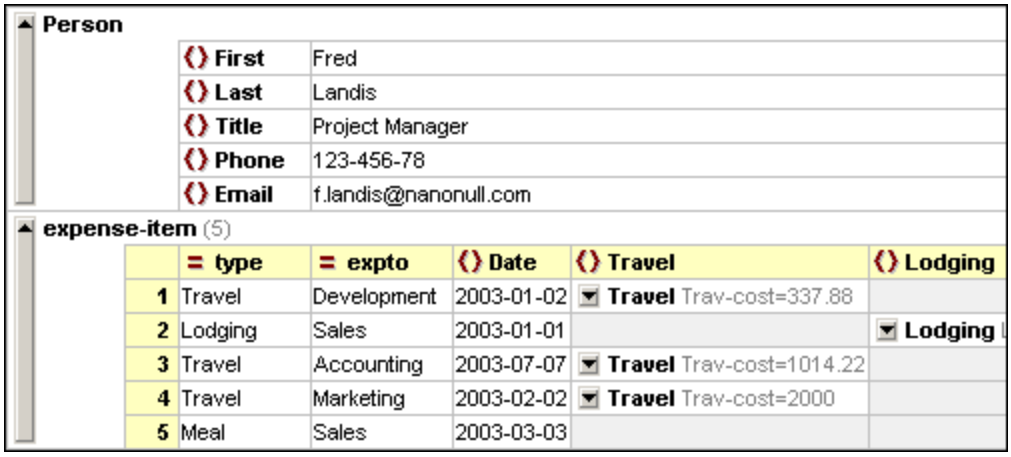
Légende :

*	Pris en charge
(1)	XSLT 2.0, XSLT 3.0 et XQuery utilisent la fonction `fn:collection`. La mise en place dans les moteurs Altova XSLT 2.0, XSLT 3.0 et XQuery résolvent des caractères génériques. D'autres moteurs peuvent se comporter différemment.

7.2.1 Exemple : Séparer un fichier XML en plusieurs fichiers

Cet exemple vous montre comment générer dynamiquement plusieurs fichiers XML depuis un seul fichier XML source. Le mappage d'accompagnement pour cet exemple est disponible sous le chemin suivant : **<Documents>\Altova\MapForce2024\MapForceExamples\Tutorial\Tut-ExpReport-dyn.mfd**.

Le fichier XML de source (disponible dans le même dossier que le mappage) contient les notes de frais d'une personne appelée "Fred Landis" et contient cinq postes de dépenses de types différents. L'objectif de cet exemple est de générer un fichier XML séparé pour chacun des postes de dépense recensé ci-dessous.



Person

First	Fred
Last	Landis
Title	Project Manager
Phone	123-456-78
Email	f.landis@nanonull.com

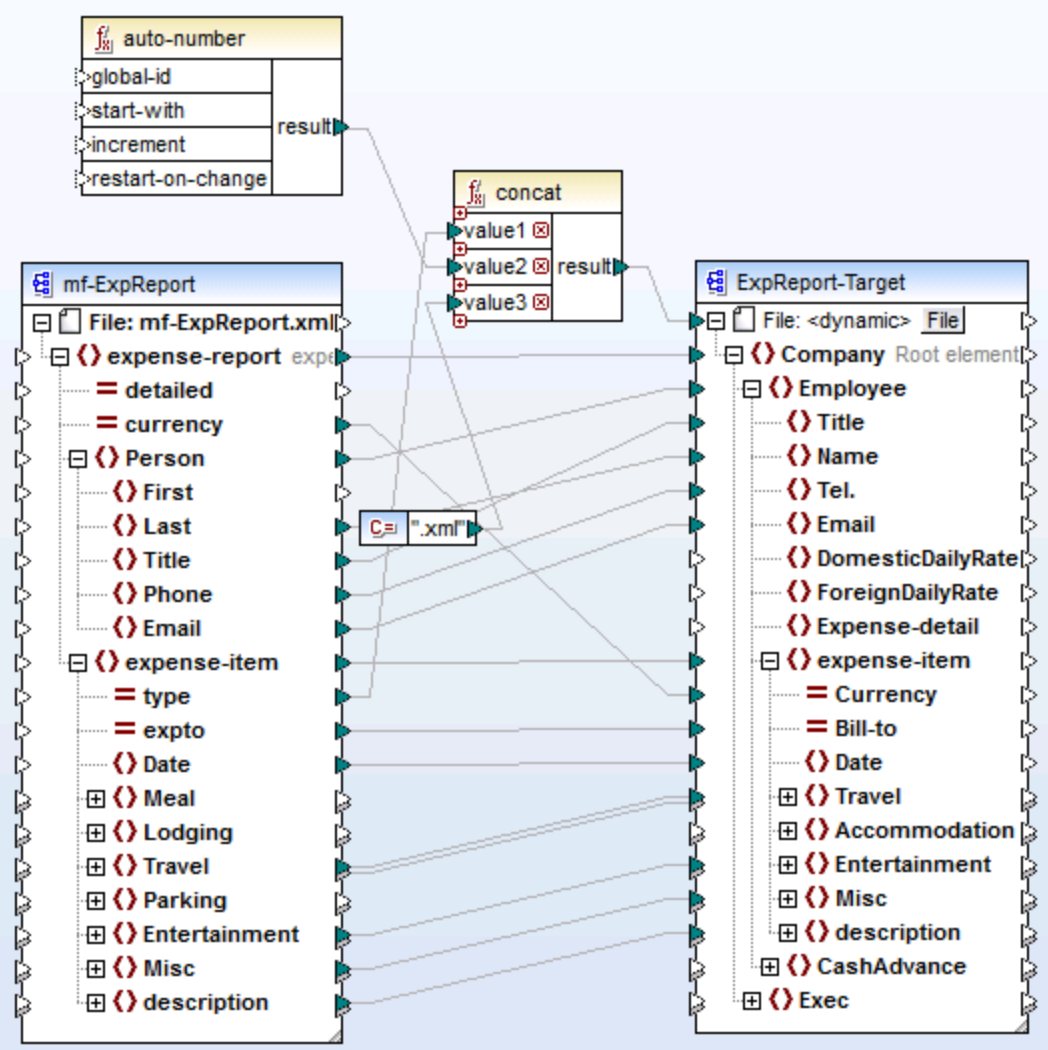
expense-item (5)

	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 (as shown in XMLSpy Grid view)

Étant donné que l'attribut `type` défini le type d'item de dépense spécifique, voici l'item que nous allons utiliser pour séparer le fichier de source. Pour atteindre l'objectif de cet exemple, procéder comme suit :

1. Insérer une fonction **`concat`** (vous pouvez la glisser depuis la bibliothèque **core | fonctions string** du volet Bibliothèques).
2. Insérer une constante (dans le menu **Insérer**, cliquer sur **Constante**) et saisir ".xml" en tant que sa valeur.
3. Insérer la fonction **`auto-number`** (vous pouvez la glisser depuis la bibliothèque **core | fonctions générateur** du volet Bibliothèques).
4. Cliquer sur la touche **Fichier** (File) ou **Fichier/String** (File/String) du composant cible et sélectionner **Utiliser les noms de fichier dynamique fournis par le mappage.**
5. Créer les connexions tel qu'affiché ci-dessous puis cliquer sur l'onglet **Sortie** pour voir le résultat du mappage.



Tut-ExpReport-dyn.mfd (MapForce Basic Edition)

Veuillez noter que les fichiers de sortie résultants sont nommés dynamiquement comme suit :

- L'attribut `type` fournit la première partie du nom de fichier (par exemple, "Travel").
- La fonction **`auto-number`** fournit le numéro séquentiel du fichier (par exemple, "Travel1", "Travel2", etc.).
- La constante fournit l'extension de fichier, qui est ".xml", donc "Travel1.xml" est le nom de fichier du premier fichier.

7.3 Règles et stratégies de mappage

En général, MapForce mappe des données de manière intuitive, mais il peut tomber sur des situations dans lesquelles la sortie contient trop ou pas assez d'items. Ce chapitre a pour but de vous aider à éviter des situations dans lesquelles le mappage produit un résultat non désiré dû à des connections incorrectes ou un contexte de mappage.

Règles de mappage

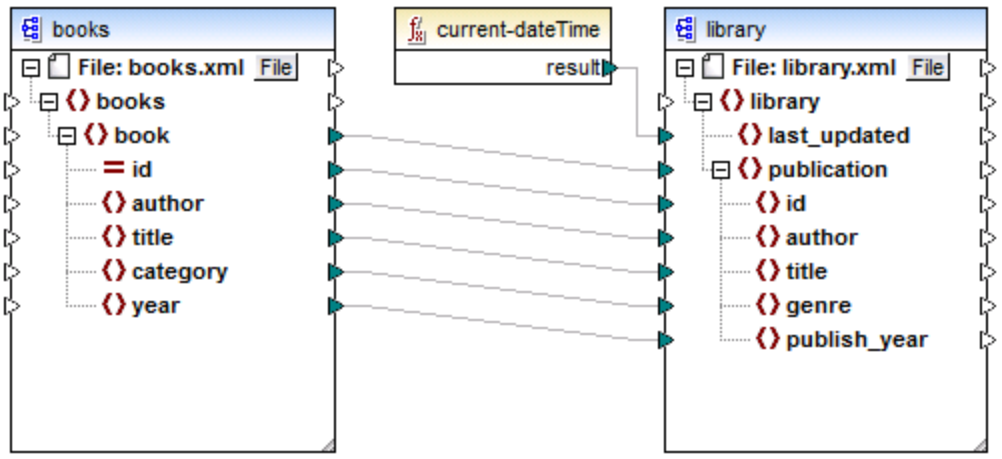
Pour être valide, un mappage doit inclure au moins une source et au moins un composant de cible. Un composant de source est un composant qui lit des données, généralement depuis un fichier ou une base de données. Un composant de cible est un composant qui écrit des données, généralement vers un fichier ou une base de données. Si vous tentez d'enregistrer un mappage où les points cités ci-dessus ne sont pas vrais, une erreur apparaît dans la fenêtre Message : "Un mappage exige au moins deux structures connectées, pour un exemple, un schéma ou une structure de base de données".

Pour créer un mappage de données, vous dessinez des connexions de mappage entre des items dans les composants de source et de cible.

Toutes les connexions de mappage que vous tracez forment, ensemble, un algorithme de mappage. Au moment de l'exécution du mappage, MapForce évalue l'algorithme et traite les données basées sur celui-ci. L'intégrité et l'efficacité de l'algorithme de mappage dépend principalement des connexions. Vous pouvez modifier certains paramètres au niveau du mappage 74, au niveau du composant 39, ou même au niveau de la connexion 49, mais, essentiellement, les *connexions* de mappage déterminent comment sont traitées vos données.

Gardez en tête les règles suivantes lorsque vous créez des connexions :

1. Lorsque vous tracez une connexion *depuis* un item de source, le mappage lit les données associées avec cet item provenant du fichier ou de la base de données source. Les données peuvent avoir zéro, une, ou plusieurs occurrences (en d'autres termes, il peut s'agir d'une séquence, comme décrit ci-dessous). Par exemple, si le mappage lit des données depuis un fichier XML contenant des livres, le fichiers XML source peut contenir zéro, un ou plusieurs éléments **book**. Dans le mappage ci-dessous, veuillez noter que l'item **book** apparaît une seule fois dans le composant de mappage, bien que le fichier de source (instance) peut contenir plusieurs éléments **book**, ou aucun.



2. Lorsque vous tracez une connexion *vers* un item de cible, le mappage génère des données d'instance de ce type. Si l'item de source contient un contenu simple (par exemple, string ou entier) et si l'item cible accepte un contenu simple, MapForce copie le contenu vers l'item de cible et, si nécessaire, convertir le type de données. Zéro, une ou plusieurs valeurs peuvent être générées, selon les données de source entrantes, voir le point suivant.
3. Pour chaque (instance) item dans la source, un (instance) item est créé dans la cible. **Il s'agit là de la règle générale de mappage dans MapForce.** En prenant le mappage ci-dessus en guise d'exemple, si le XML source contient trois éléments **book**, alors trois éléments **publication** seront créés du côté cible. Veuillez noter qu'il existe aussi des cas spéciaux, voir Séquences 406.
4. Chaque connexion crée un *contexte de mappage actuel*. Le contexte détermine quelles données sont disponibles *actuellement, pour le nœud de cible actuel*. Ainsi, le contexte détermine quels items de source sont réellement copiés depuis la source vers le composant de cible. En traçant ou en omettant une connexion, vous pouvez modifier par inadvertance le contexte actuel et influer le résultat du mappage. Par exemple votre mappage peut appeler une base de données ou un service Web plusieurs fois sans que cela soit nécessaire au cours d'une seule exécution de mappage. Ce concept est décrit plus en détail ci-dessous, voir Le contexte de mappage 407.

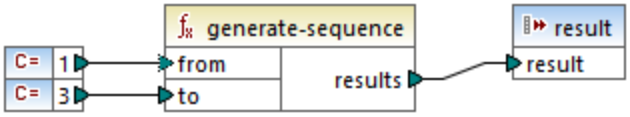
7.3.1 Séquences

Comme mentionné auparavant, la règle de mappage générale est : "pour chaque item se trouvant dans la source, en créer un dans la cible". Ici, "item" signifie une des choses suivantes :

- un nœud unique d'instance du fichier ou de la base de données d'entrée
- une séquence de zéro à plusieurs nœuds d'instance du fichier ou de la base de données d'entrée

Pendant l'exécution de mappage, si une séquence atteint un item de cible, cela crée une boucle qui génère autant de nœuds cible qu'il y a de nœuds de source. Néanmoins, il y a quelques exceptions à cette règle :

- Si l'item de cible est un élément root XML, il est créé une fois (et une seule fois). Si vous y connectez une séquence, le résultat peut ne pas être valide pour le schéma. Si les attributs de l'élément root sont aussi connectés, la sérialisation XML échouera au moment de l'exécution du mappage. C'est pourquoi, il est nécessaire de connecter une séquence à l'élément XML root.
- Si l'item de cible accepte uniquement une valeur, elle est créée une seule fois. Des exemples d'items qui acceptent uniquement une seule valeur : Attributs XML, champs de base de données, composants de sortie simples. Par exemple, le mappage ci-dessous génère une séquence de trois entiers (1, 2, 3) avec l'aide de la fonction `generate-sequence`. Néanmoins, le résultat contiendra uniquement un seul entier, cas la cible est un seul composant de résultat qui accepte une seule valeur. Les deux autres valeurs sont ignorées.



- Si le schéma de source précise qu'un item spécifique ne se produit qu'une seule fois, mais que le fichier d'instance en contient de nombreux, MapForce peut extraire le premier item depuis la source (qui doit exister conformément au schéma) et créer un seul item dans la cible. Pour désactiver ce comportement, décocher la case **activer les optimisation de traitement d'entrée sur la base de min/maxOccurs** depuis les paramètres de composant, voir aussi Paramètres de composant XML 113.

Si la séquence est vide, rien n'est généré du côté cible. Par exemple, si la cible est un document XML et que la séquence de source est vide, aucun élément XML ne sera créé dans la cible.

Les fonctions agissent de manière similaire : si elles obtiennent une séquence en tant qu'entrée, alors elles seront appelées (et produire autant de résultats que) autant de fois qu'il y a d'items dans la séquence.

> Si une fonction obtient une séquence vide en tant qu'entrée, elle retourne un résultat vide également, par conséquent, elle ne produit aucune sortie.

Néanmoins, il existe certaines catégories de fonctions qui, en raison de leur design, retournent une valeur même si elle obtiennent une séquence vide en tant qu'entrée :

- **`exists`**, **`not-exists`**, **`substitute-missing`**
- **`is-null`**, **`is-not-null`**, **`substitute-null`** (ces trois fonctions sont des alias des trois précédents)
- fonctions aggregate (**`sum`**, **`count`**, etc.)
- Fonctions définies par l'utilisateur qui acceptent des séquences et sont des fonctions régulières (pas inlined)

Si vous souhaitez remplacer une valeur vide, ajouter la fonction **`substitute-missing`** au mappage et remplacer la valeur vide par une valeur de substitution de votre choix.

Les fonctions peuvent avoir plusieurs entrées. Si une séquence est connectée à chaque entrée, cela résulte en un produit cartésien de toutes les entrées, ce qui n'est généralement pas le résultat souhaité. Afin d'éviter cela, ne connecter qu'une seule séquence à une fonction avec plusieurs paramètres ; tous les autres paramètres doivent être connectés aux items "singular" provenant de parents ou d'autres composants.

7.3.2 Le contexte de mappage

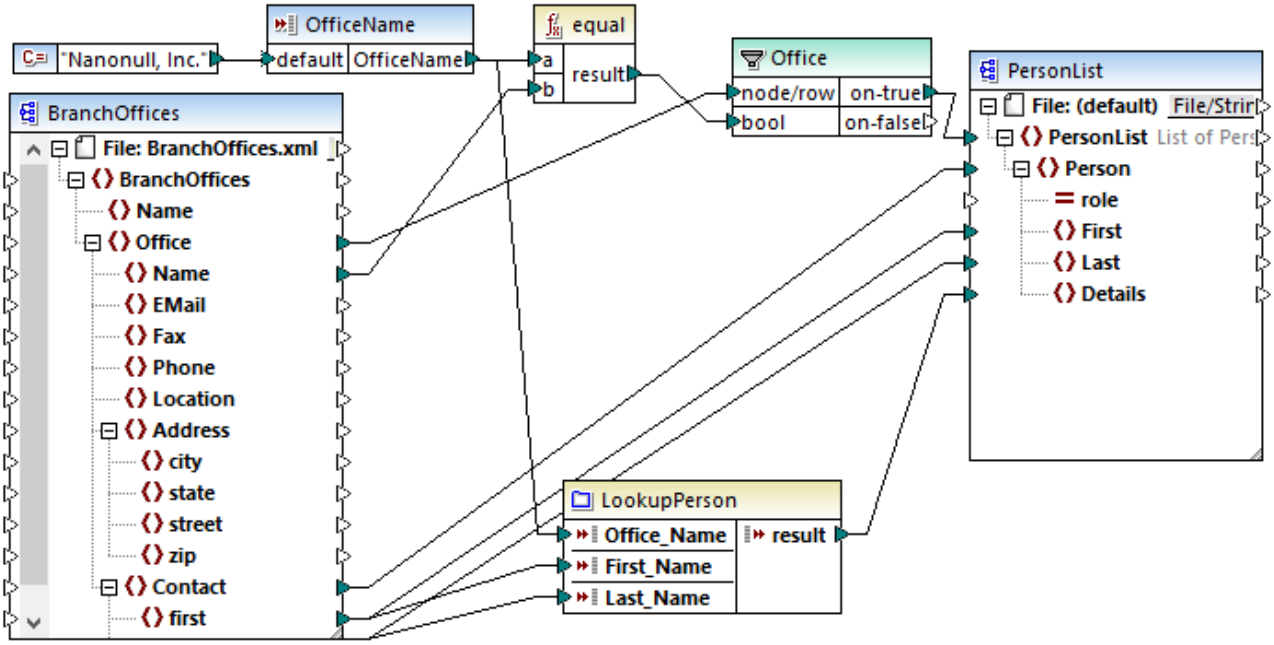
Les composants de mappage sont des structures hiérarchiques qui peuvent contenir plusieurs niveaux de profondeur. D'un autre côté, un mappage peut avoir plusieurs sources et composants, plus des composants intermédiaires comme des fonctions, des filtres, des value-maps, etc. Cela rajoute de la complexité dans l'algorithme de mappage, en particulier si plusieurs composants non-liés sont connectés. Afin de pouvoir exécuter le mappage en petites parties, une étape à la fois, un contexte actuel doit être établi poru chaque connexion.

On pourrait donc dire que plusieurs "contextes actuels" sont établis pour la durée de l'exécution de mappage, depuis les modifications de contexte actuels à chaque connexion traitée.

MapForce établit toujours le contexte actuel en commençant depuis l'*item de racine cible (nœud)*. C'est là que l'exécution du mappage commence réellement. La connexion à l'item de racine cible est retracée à tous les items de source qui y sont connectés directement ou indirectement, y compris par le biais de fonctions ou d'autres composants intermédiaires. Tous les items de source et les résultats produits par les fonctions sont ajoutés au contexte actuel.

Une fois que le processus dans le nœud cible est terminé, MapForce se déplace vers le bas de la hiérarchie. Concrètement, il traite tous les *items mappés* du composant cible du haut en bas. Pour chaque item, un nouveau contexte est établi qui contient au début tous les items du contexte parent. Ainsi, tous les items frères mappés dans un composant cible sont indépendants l'un de l'autre, mais ont accès à toutes les données de source de leurs items parent.

Nous allons voir comment les éléments sus-mentionnés s'appliquent en pratique, sur la base d'un mappage d'exemple, **PersonListByBranchOffice.mfd**. Vous trouverez ce mappage dans le répertoire **<Documents>\Altova\MapForce2024\MapForceExamples**.



Dans le mappage ci-dessus, les composants de source et cible sont XML. Le fichier de source XML contient deux éléments **Office**.

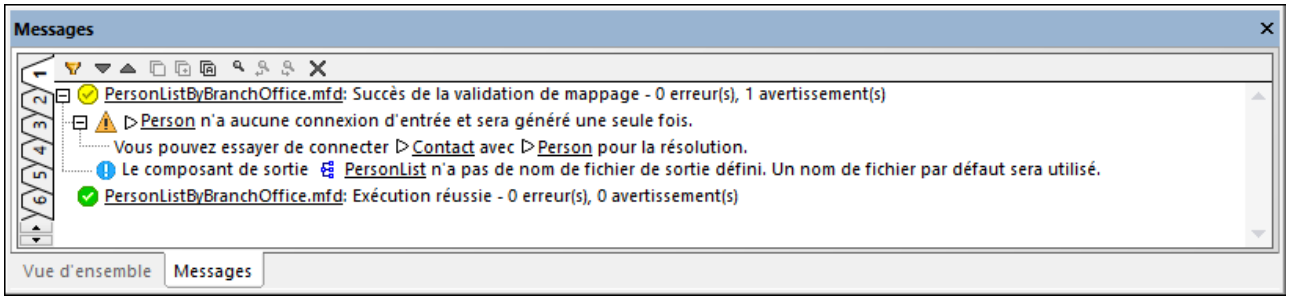
Comme mentionné précédemment, l'exécution de mappage commence toujours depuis le nœud de racine cible (**PersonList**, dans cet exemple). En retraçant la connexion (par le biais du filtre et de la fonction) vers un item de source, vous pouvez conclure que l'item de source est **Office**. (L'autre chemin de connexion mène à un paramètres d'entrée et son objectif est expliqué ci-dessous).

S'il y avait une connexion directe entre **Office** et **PersonList**, alors, conformément à la règle de mappage générale, le mappage aurait créé autant d'items d'instance **PersonList** qu'il y a d'items **Office** dans le fichier source. Néanmoins, cela ne se produira pas ici, étant donné qu'il y a un filtre entre les deux. Le filtre n'apporte au composant de cible que les données qui satisfont à la condition Booléenne connectée à l'entrée **bool** du filtre. La fonction **equal** retourne **true** si le nom de bureau est égal à "Nanonull, Inc.". Cette condition n'est satisfaite qu'une seule fois, car il n'y a qu'un seul nom de bureau dans le fichier XML de source.

Par conséquent, la connexion entre **Office** et **PersonList** définit un seul bureau en tant que le contexte pour l'ensemble du document cible. Cela signifie que tous les descendants de l'item **PersonList** ont accès aux données du bureau "Nanonull, Inc.", et qu'aucun autre bureau n'existe dans le contexte actuel.

La connexion suivante est entre **Contact** et **Person**. Conformément à la règle générale de mappage, elle créera une cible **Person** pour chaque source **Contact**. À chaque itération, cette connexion établit un nouveau contexte actuel ; c'est pourquoi les connexions enfant (**first** à **First**, **last** à **Last**) fournissent des données depuis l'item de source vers la cible dans le contexte de chaque **Person**.

Si vous avez ignoré la connexion entre **Contact** et **Person**, alors le mappage créera uniquement une **Person** avec plusieurs nœuds **First**, **Last**, et **Details**. Dans ces cas, MapForce émet un avertissement et une suggestion dans la fenêtre Messages, par exemple :



Enfin, le mappage comprend une fonction définie par l'utilisateur, **LookupPerson**. La fonction définie par l'utilisateur est aussi exécutée dans le contexte de chaque **Person**, étant donné que la connexion parent entre **Contact** et **Person**. Spécifiquement, à chaque fois qu'un nouvel item Person est créé du côté cible, la fonction est appelée pour remplir l'élément **Details** de la personne. Cette fonction prend trois paramètres d'entrée. Le premier (**OfficeName**) est défini pour lire des données provenant du paramètres d'entrée du mappage. Les données source pour ce paramètre peuvent aussi bien être fournies par l'item de source **Name**, sans aucunement changer la sortie de mappage. Dans tous les cas, la valeur de source est la même et est prise depuis un contexte de parent. En interne, la fonction de consultation concatène les valeurs reçues en tant qu'arguments et produit une seule valeur. Pour plus d'informations concernant le fonctionnement de la fonction **LookupPerson**, voir l'Exemple: Consultation et Concaténation 216.

7.3.2.1 Fonctions définies par l'utilisateur

Les fonctions définies par l'utilisateur (FDUs) sont des fonctions personnalisées intégrées dans le mappage, où vous définissez les entrées, les sorties et la logique de traitement. Chaque fonction définie par l'utilisateur peut contenir le même type de composant en tant que mappage principal, y compris les services Web et les bases de données.

> Par défaut, si une FDU contient une base de données ou un composant de service Web et si les données d'entrée du FDU est une séquence de plusieurs valeurs, chaque entrée de valeur appellera la FDU et par conséquent entraînera dans une base de données ou un appel de service Web.

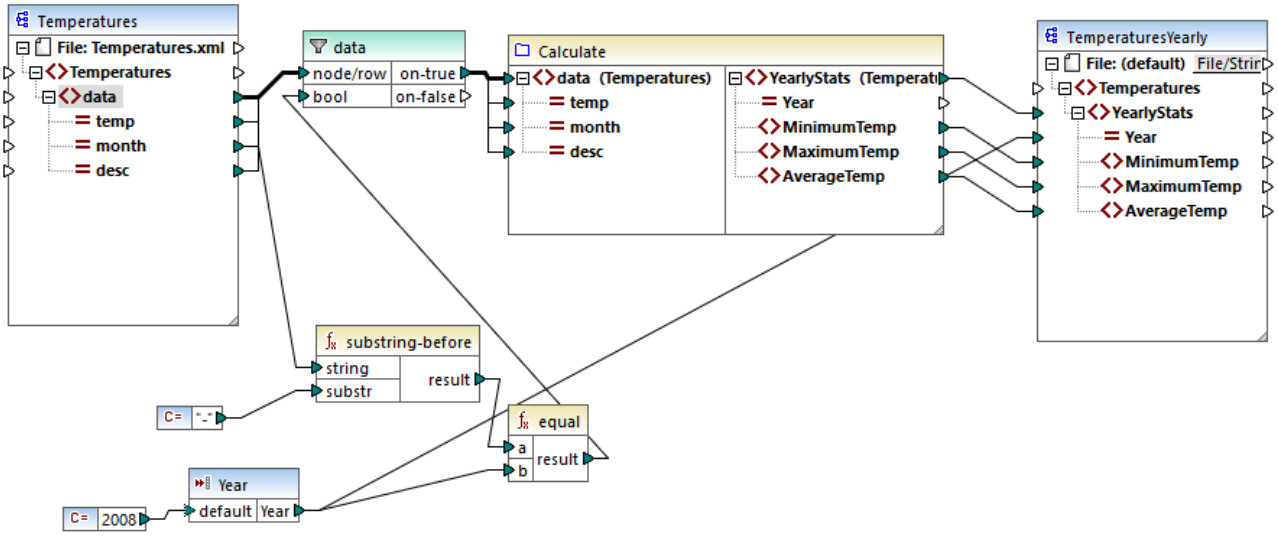
Ce comportement mentionné peut être acceptable pour le type de mappage dans lequel vous avez vraiment besoin que la FDU soit appelée *autant de fois qu'il y a de valeurs d'entrée* et que vous n'avez pas d'alternative.

Si vous ne souhaitez pas que cette situation se produise, vous pouvez configurer la FDU de manière à ce qu'elle soit appelée une seule fois même si elle reçoit une séquence de valeurs en tant qu'entrée. Généralement, c'est la technique que vous adopterez pour les FDU qui fonctionnement sur un ensemble de valeurs avant qu'elles puissent être retournées (comme des fonctions qui calculent les moyennes ou les totaux).

Configurer une FDU pour accepter plusieurs valeurs d'entrée dans le même appel de fonction est possible si la FDU est de type "regular", pas "inlined". (Pour plus de détails, voir le chapitre Fonctions définies par l'utilisateur 203.) Avec des fonctions régulières, vous pouvez spécifier que le paramètres d'entrée est une séquence en cochant la case **Entrée est une séquence**. Cette case à cocher est visible dans les paramètres de composant, après avoir double-cliqué dans la barre de titre d'un paramètre d'entrée. La case à cocher influe sur le nombre de fois que la fonction est appelée :

- Lorsque les données d'entrée sont connectées à un paramètre **sequence**, la fonction définie par l'utilisateur est appelée *uniquement une fois* et la séquence complète est passée dans la fonction définie par l'utilisateur.
- Lorsque les données d'entrée sont connectées à un paramètre **non-sequence**, la fonction définie par l'utilisateur est appelée *une seule fois dans chaque item unique dans la séquence*.

Pour un exemple, ouvrir le mappage de démonstration suivant :
<Documents>\Altova\MapForce2024\MapForceExamples\InputIsSequence.mfd.

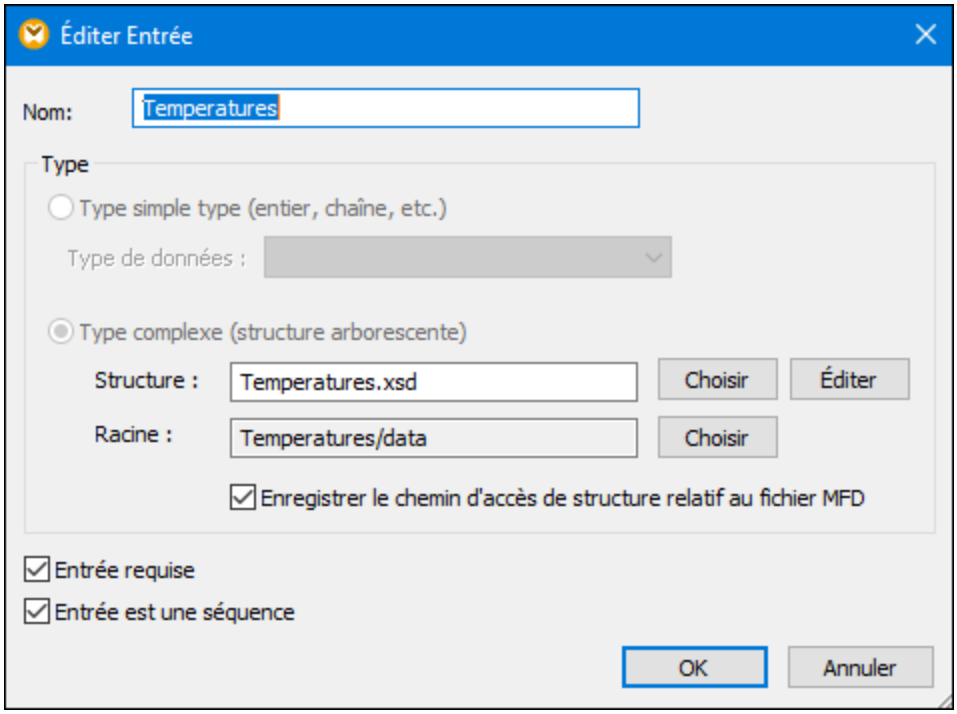


Le mappage ci-dessus illustre un cas typique d'une FDU qui fonctionne sur un ensemble de valeurs et nécessite donc toutes les valeurs d'entrée en un appel. Spécifiquement, la fonction définie par l'utilisateur **`Calculate`** retourne les températures minimum, maximum et moyennes, en prenant en tant que données d'entrée provenant d'un fichier XML de source. La sortie de mappage attendue est comme suit :

```
<Temperatures>
    <YearlyStats Year="2008">
        <MinimumTemp>-0.5</MinimumTemp>
        <MaximumTemp>24</MaximumTemp>
        <AverageTemp>11.6</AverageTemp>
    </YearlyStats>
</Temperatures>
```

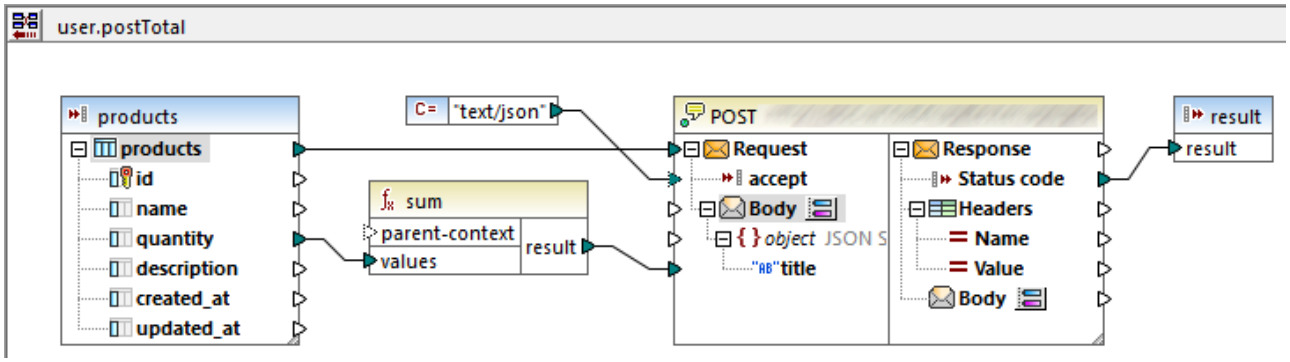
Comme d'habitude, l'exécution de mappage commence avec l'item supérieur du composant cible (**YearlyStats**, dans cette exemple). Pour remplir ce nœud, le mappage tente d'obtenir des données de source provenant de la FDU, qui, en revanche, déclenche le filtre. Le rôle du filtre dans ce mappage est de passer dans la FDU uniquement les températures depuis l'année 2008.

La case à cocher **Entrée est séquence** a été cochée pour le paramètre d'entrée de la FDU (pour voir cette case à cocher, double-cliquer sur la barre de titre de la fonction **`Calculate`** pour saisir le mappage de la fonction ; ensuite double-cliquer sur la barre de titre du paramètre d'entrée). Comme évoqué plus haut, l'option **Entrée est séquence** option entraîne l'approvisionnement de la séquence de valeurs complète en tant qu'entrée dans la fonction et la fonction est appelée une seule fois.



Si vous n'aviez pas coché la case à cocher **Entrée est séquence**, la FDU aurait été appelée pour chaque valeur dans la source. Par conséquence, le minimum, le maximum et la moyenne serait calculée pour chaque valeur unique individuelle et une sortie incorrecte serait produite.

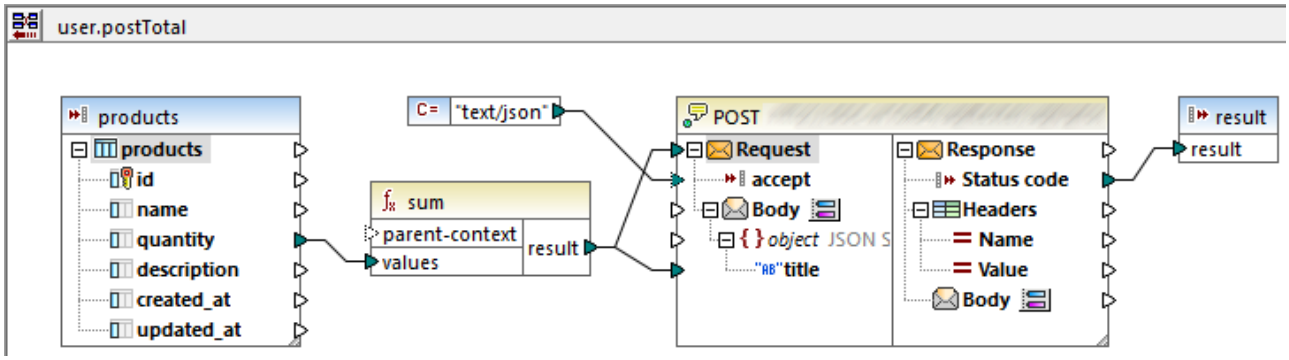
En appliquant la même logique dans des FDU plus complexes qui comprennent une base de données ou des appels de service Web, il peut être possible d'optimiser l'exécution et d'éviter des appels inutiles à la base de données ou au service Web. Néanmoins, sachez que la case à cocher **Entrée est séquence** ne contrôle pas ce qui arrive à la séquence de valeur *après* qu'elle saisit la fonction. En d'autres termes, rien ne vous empêche de connecter la séquence des valeurs entrantes dans l'entrée d'un service Web et donc de l'appeler plusieurs fois. Considérez l'exemple suivant :



La FDU illustrée ci-dessus reçoit une séquence des valeurs depuis le mappage externe. En particulier, les données fournies au paramètre d'entrée provient d'une base de données. L'option paramètre d'entrée **Entrée est séquence** est sélectionnée, donc toute la séquence est fournie à la fonction en un appel. La fonction est censée accumuler plusieurs valeurs **quantity** et publier le résultat dans un service Web. Exactement un appel de service Web est attendu. Néanmoins, le service Web sera appelé plusieurs fois de manière incorrecte

lorsque le mappage est exécuté. La raison est que l'entrée **Request** du service Web reçoit *une séquence de valeurs*, pas une seule valeur.

Pour régler ce problème, connecter l'entrée **Request** du service Web dans le résultat de la fonction **sum**. La fonction produit une seule valeur, donc le service Web sera aussi appelé une fois :



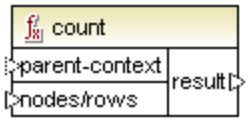
Normalement, les fonctions aggregate comme **sum**, **count**, etc produisent une seule valeur. Néanmoins, s'il existe une connexion parent qui le permet, elles peuvent également produire une séquence de valeurs, comme décrit plus loin dans l'Exemple : Changer le contexte Parent 412.

7.3.2.2 Exemple : Changer le contexte de parent

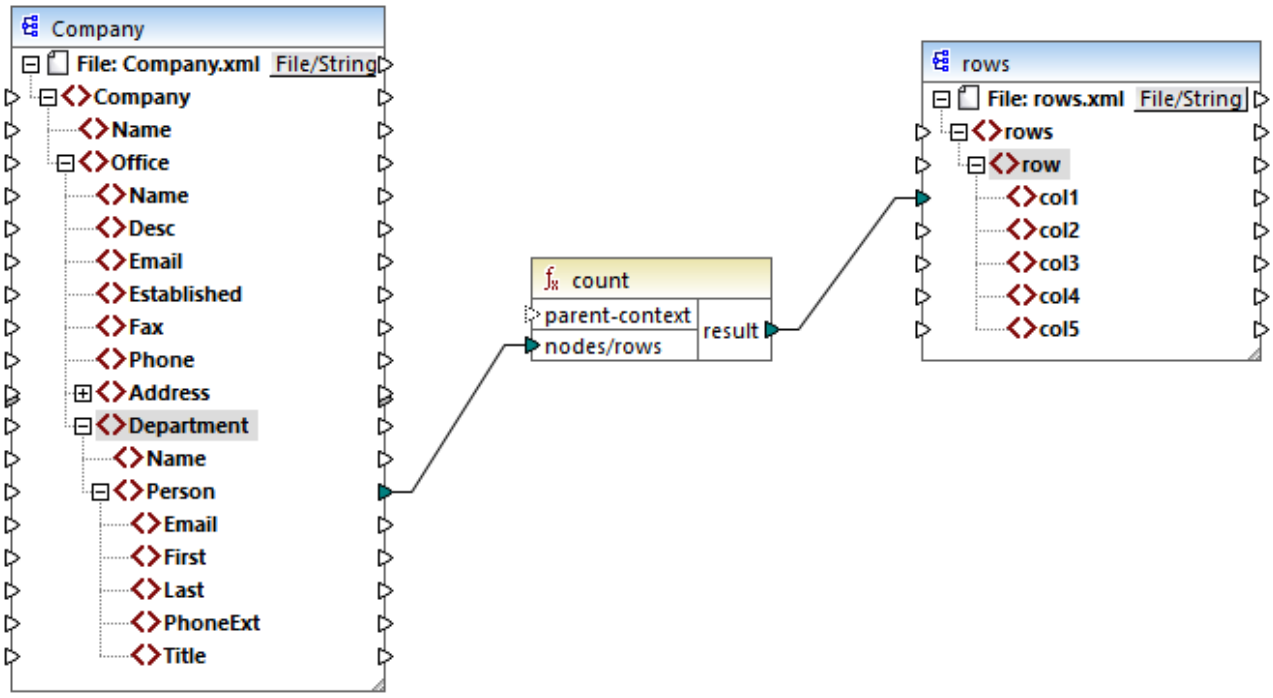
Certains composants de mappage ont un item optionnel **parent-context**.

`parent-context` est un argument optionnel dans certaines fonctions d'agrégation core MapForce (comme dans **min**, **max**, **avg**, **count**). Dans un composant de source qui possède plusieurs séquences hiérarchiques, le contexte parent détermine l'ensemble de nœuds dans lequel la fonction doit fonctionner.

À l'aide de cet item, vous pouvez influer sur le mappage dans lequel ce composant devrait opérer et par conséquent changer le résultat du mappage. Les composants qui ont un **parent-context** optionnel sont : fonctions aggregate, variables et composants Join.



Pour voir comment la modification du contexte de parent peut être utile, consulter une démo en ouvrant le mappage suivant :
<Documents>\Altova\MapForce2024\MapForceExamples\Tutorial\ParentContext.mfd.



Dans le XML source du mappage ci-dessus, il y a un nœud **Company** unique qui contient deux nœud **Office**. Chaque nœud **Office** contient plusieurs nœuds **Department**, et chaque **Department** contient plusieurs nœud **Person**. Si vous ouvrez le fichier XML dans un éditeur XML, vous pouvez voir que la distribution des personnes par bureau et par département est la suivante :

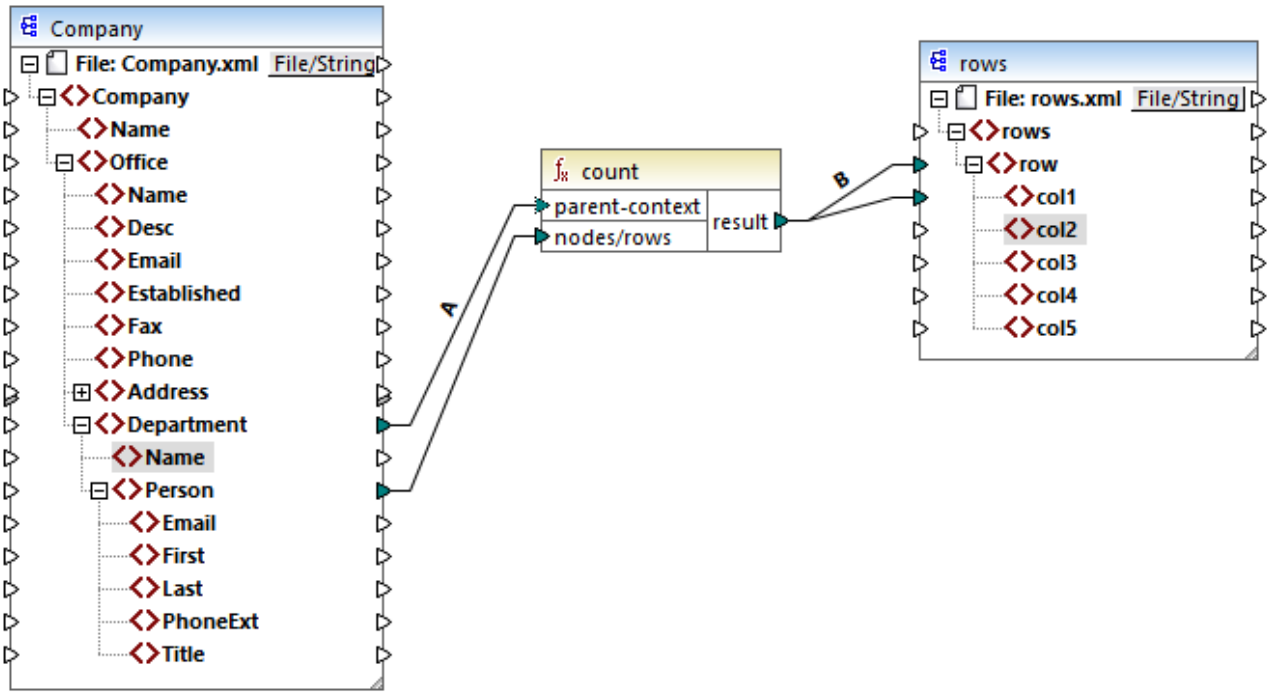
Bureau	Département	Nombre d'employés
Nanonull, Inc.	Administration	3
	Marketing	2
	Engineering	6
	IT & Technical Support	4
Nanonull Partners, Inc.	Administration	2
	Marketing	1
	IT & Technical Support	3

Le mappage compte toutes les personnes dans tous les départements. À cet effet, il utilise la fonction **count** depuis la bibliothèque `core`. Si vous cliquez sur l'onglet **Output** pour consulter le mappage, vous remarquerez qu'il produit une seule valeur, **21**, qui correspond au nombre total de personnes contenues dans le fichier XML source.

Le mappage fonctionne comme suit:

- Comme d'habitude, l'exécution de mappage débute avec le nœud supérieur du composant cible (**rows**, dans cet exemple). Il n'y a pas de connexion entrante pour **rows**. En résultat, un contexte de mappage implicite est établi entre **Company** (item supérieur du composant de source) et **rows** (item supérieur du composant cible).
- Le résultat de la fonction est une seule valeur, puisqu'il n'y a qu'une seule entreprise dans le fichier source.
- Afin de remplir l'item cible **col1**, MapForce exécute la fonction **count** dans le *contexte de parent implicite* mentionné ci-dessus, afin qu'il compte tous les nœuds **Person** depuis tous les bureaux et tous les départements.

L'argument **parent-context** de la fonction vous permet de modifier le contexte de mappage. Cela vous permet, par exemple, de compter le nombre de personnes dans chaque département. Pour ce faire, tracer deux (ou plus) connexions comme indiqué ci-dessous :



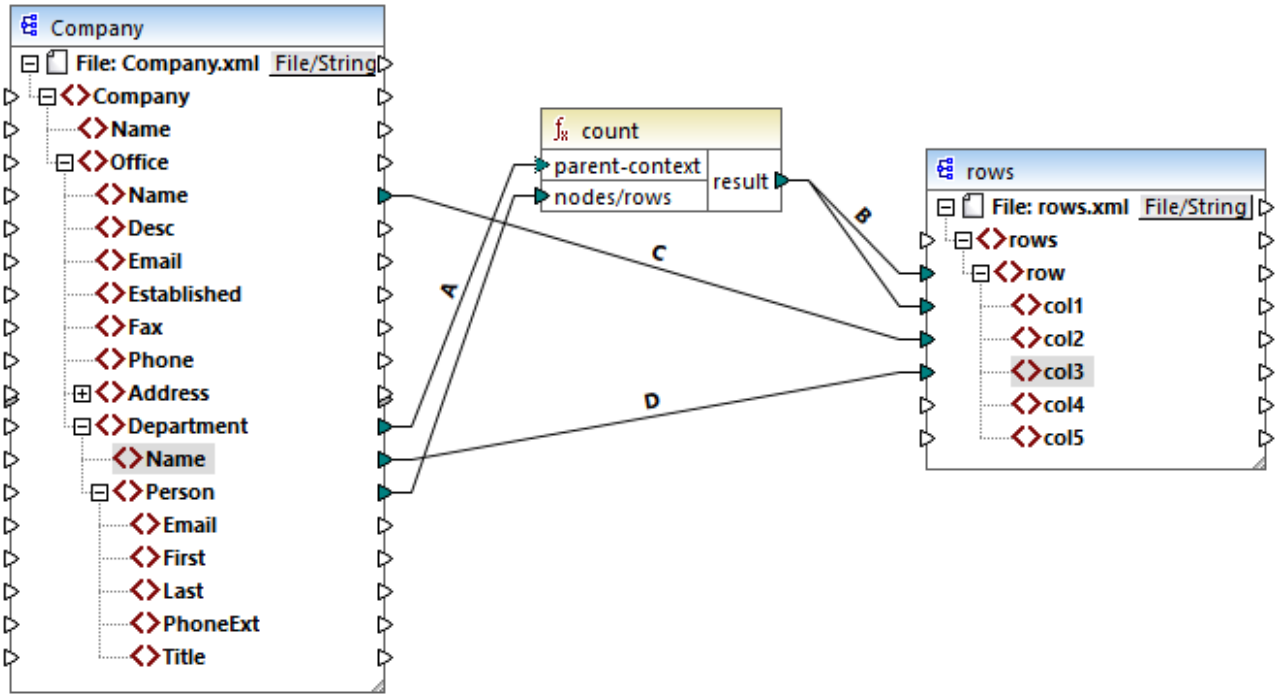
Dans le mappage ci-dessus, la connexion A change le contexte de parent de la fonction **count** dans le **Department**. En résultat, la fonction comptera le nombre de personnes se trouvant dans chaque département. Chose importante, la fonction retournera maintenant une *séquence* de résultats au lieu d'un seul résultat, étant donné que plusieurs département existent dans la source. C'est la raison pour laquelle la connexion B existe: pour chaque item dans la séquence résultante, elle crée une nouvelle ligne dans le fichier cible. Le résultat de mappage change maintenant (veuillez noter que les nombres correspondent exactement au décompte des personnes dans chaque département) :

```
<rows>
    <row>
        <col1>3</col1>
    </row>
    <row>
        <col1>2</col1>
```



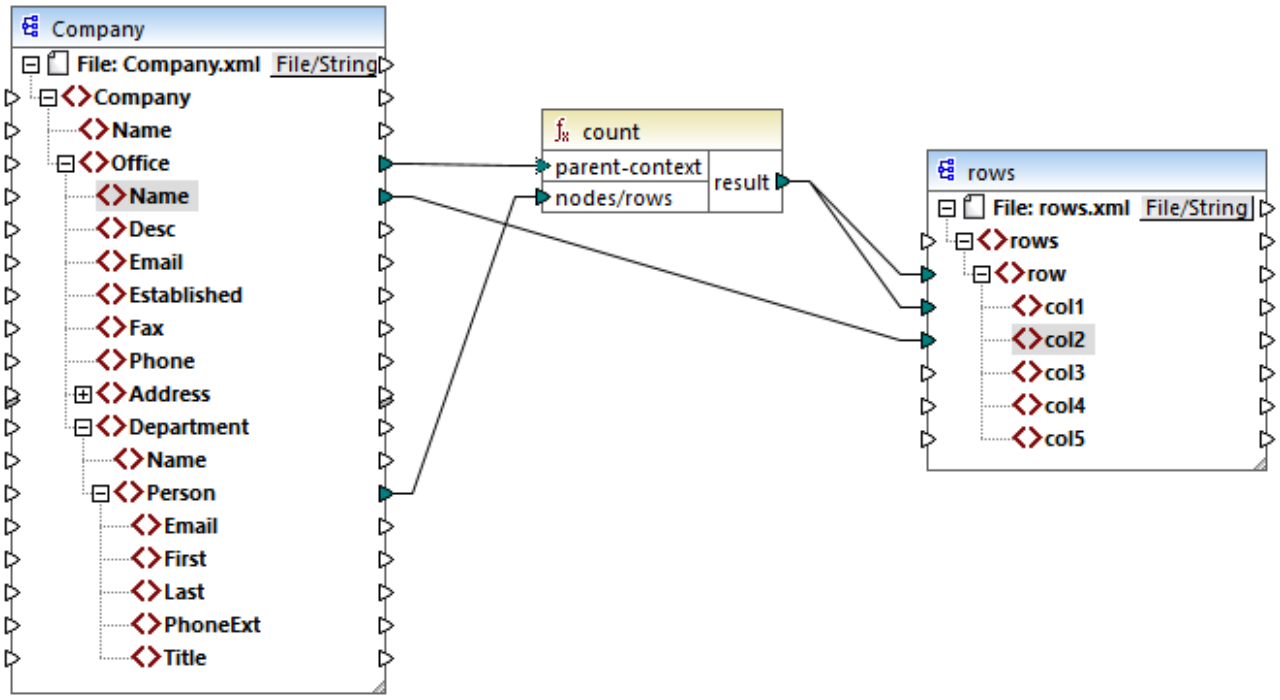
```
    </row>
    <row>
        <col1>6</col1>
    </row>
    <row>
        <col1>4</col1>
    </row>
    <row>
        <col1>2</col1>
    </row>
    <row>
        <col1>1</col1>
    </row>
    <row>
        <col1>3</col1>
    </row>
</rows>
```

Étant donné que le mappage actuel crée une ligne dans chaque département, vous pouvez copier en option le nom du bureau et du département aussi dans le fichier cible, en traçant des connexions C et D :



Ainsi, le résultat affichera non seulement le nombre de personnes mais aussi le nom correspondant du bureau et du département.

Si vous souhaitez compter le nombre de personnes dans chaque bureau, connectez le contexte de parent de la fonction **count** à l'item **Office** dans la source.



Avec les connexions affichées ci-dessus, la fonction **count** retourne un résultat pour chaque bureau. Il y a deux bureaux dans le fichier source, la fonction retournera maintenant deux séquences. Par conséquent, il y aura deux lignes dans le résultat, chaque ligne contiendra le nombre de personnes contenues dans ce bureau :

```
<rows>
    <row>
        <col1>15</col1>
        <col2>Nanonull, Inc.</col2>
    </row>
    <row>
        <col1>6</col1>
        <col2>Nanonull Partners, Inc.</col2>
    </row>
</rows>
```

7.3.3 Contexte de priorité

Le contexte de priorité est une méthode d'influencer l'ordre dans lequel les paramètres d'entrée d'une fonction sont évalués. La configuration d'un contexte de priorité peut être nécessaire si votre mappage rejoint des données provenant de deux sources non liées.

Afin de comprendre comment le contexte de priorité fonctionne, souvenez-vous que, lorsqu'un mappage est exécuté, la connexion à un item d'entrée peut porter une *séquence* de plusieurs valeurs. En ce qui concerne les fonctions avec deux paramètres d'entrée, cela signifie que MapForce doit créer deux boucles, dont une doit être traitée en premier. Cette boucle est la boucle "extérieure". Par exemple, la fonction **equal** reçoit deux

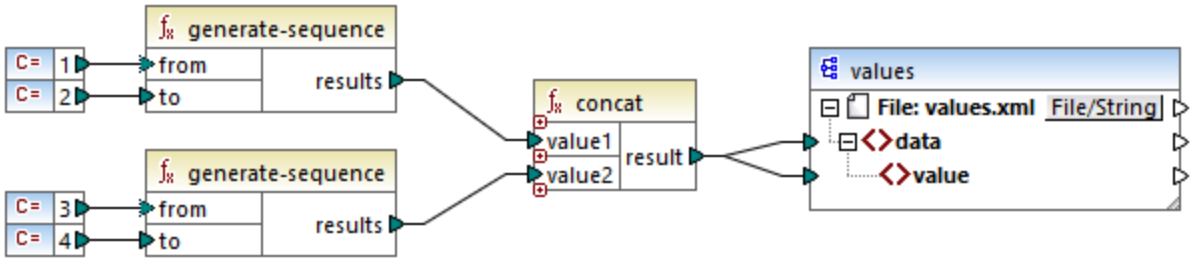
paramètres : *a* et *b*. si *a* et *b* obtiennent tous deux une séquence de valeurs, alors MapForce traite comme suit :

- Pour chaque occurrence de *a*
  - Pour chaque occurrence de *b*
    - *a* est-il égal à *b*?

Comme vous pouvez le voir dans l'exemple ci-dessus, chaque *b* est évalué dans le contexte de chaque *a*. Le contexte de priorité vous permet de modifier la logique de traitement de manière à ce que chaque *a* est évalué dans le contexte de chaque *b*. En d'autres termes, il vous permet d'échanger la boucle intérieure avec la boucle extérieure, par exemple :

- Pour chaque occurrence de *b*
  - Pour chaque occurrence de *a*
    - *a* est-il égal à *b*?

Examinons à présent un mappage dans lequel le contexte de priorité touche la sortie du mappage. Dans le mappage ci-dessous, la fonction **concat** a deux paramètres d'entrée. Chaque paramètre d'entrée est une séquence qui a été générée avec l'aide de la fonction **generate-sequence**. La première séquence est "1,2" et la seconde séquence est "3,4".



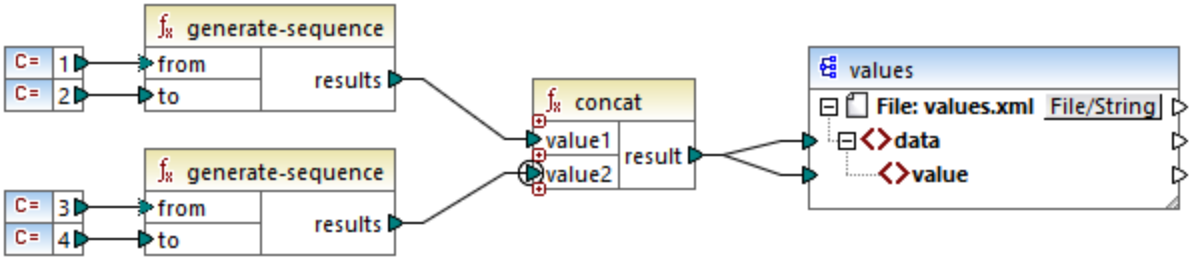
Tout d'abord, examinons le mappage sans déterminer de contexte de priorité. La fonction **concat** commence à évaluer la première séquence en premier, donc elle combine des valeurs dans l'ordre suivant :

- 1 avec 3
- 1 avec 4
- 2 avec 3
- 2 avec 4

Cela est également reflété dans le résultat de mappage :

```
<data>
    <value>13</value>
    <value>14</value>
    <value>23</value>
    <value>24</value>
</data>
```

Si vous cliquez avec la touche de droite dans le deuxième paramètre d'entrée et que vous choisissez **Contexte de priorité** depuis le menu contextuel, celui-ci deviendra le contexte de priorité. Comme illustré ci-dessous, l'entrée du contexte de priorité est encerclé.



Cette fois, le deuxième paramètre d'entrée sera évalué en premier. La fonction `concat` concatènera toujours les mêmes valeurs, mais cette fois elle traitera la séquence `3,4` en premier. Par conséquent, le résultat donne :

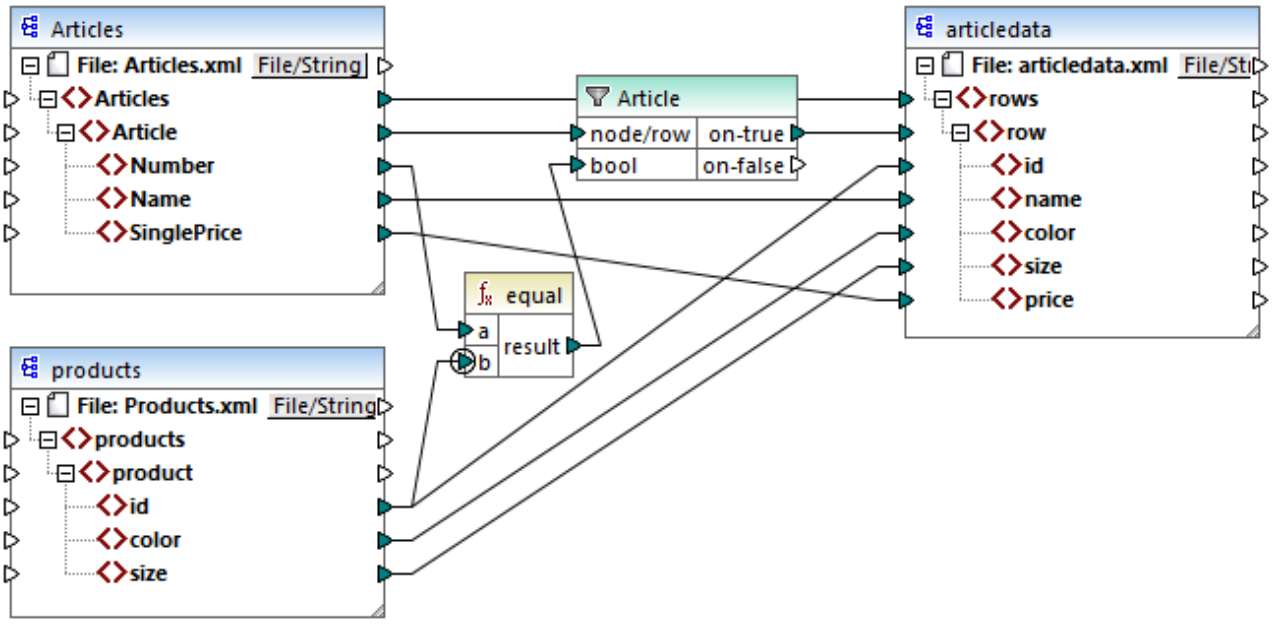
```
<data>
    <value>13</value>
    <value>23</value>
    <value>14</value>
    <value>24</value>
</data>
```

Jusqu'à présent, vous n'avez vu que la partie théorique derrière le contexte de priorité. Pour consulter un scénario plus pratique, voir Exemple: Filtrer avec un contexte de priorité 418.

7.3.3.1 Exemple: Filtrer avec un contexte de priorité

Lorsqu'une fonction est connectée à un filtre, le contexte de priorité n'affecte non seulement la fonction elle-même, mais aussi l'évaluation du filtre. Le mappage ci-dessous illustre un cas typique lorsque vous nécessitez de définir un contexte de priorité pour pouvoir obtenir le résultat correct. Vous trouverez ce mappage sous le chemin suivant :
<Documents>\Altova\MapForce2024\MapForceExamples\Tutorial\FilterWithPriority.mfd.

Note : Ce mappage utilise des composants XML, mais la même logique que celle décrite ci-dessous s'applique pour tous les autres types de composants dans MapForce, y compris EDI, JSON, etc.



L'objectif du mappage ci-dessus est de copier des données de personnes depuis **Articles.xml** dans un nouveau fichier XML avec un schéma différent, **articledata.xml**. Dans le même temps, le mappage doit consulter les détails de chaque article dans le fichier **Products.xml** et les joindre à l'enregistrement d'article respectif. Veuillez noter que chaque enregistrement dans **Articles.xml** ait un **Number** et que chaque enregistrement dans **Products.xml** ait un **id**. Si ces deux valeurs sont égales, alors toutes les autres valeurs (**Name**, **SinglePrice**, **color**, **size**) doivent être copiées dans la même **ligne** dans la cible.

Cet objectif a été accompli en ajoutant un filtre. Chaque filtre nécessite une condition Booléenne en tant qu'entrée ; seuls les nœud/lignes qui satisfont la condition seront copiés dans la cible. À cet effet, il existe une fonction `equal` dans le mappage. La fonction `equal` vérifie si le nombre d'article et l'ID de produit sont égaux dans les deux sources. Le résultat est ensuite fourni en tant qu'entrée dans le filtre. Si **true**, alors l'item **Article** est copié dans la cible.

Veuillez noter qu'un contexte de priorité a été défini dans le deuxième paramètres d'entrée de la second fonction `equal`. Dans ce mappage, le contexte de priorité fait la grande différence, et le fait de ne pas le configurer entraînera un résultat de mappage incorrect.

Mappage initial : Aucun contexte de priorité

Voici la logique de mappage sans contexte de priorité :

- Conformément à la règle de mappage général, pour chaque **Article** qui satisfait la condition de filtre, une nouvelle **row** est créée dans la cible. La connexion entre **Article** et **row** (par le biais de la fonction et du filtre) se charge de cette partie.
- Le filtre vérifie la condition pour chaque article. Pour ce faire, il itère à travers tous les produits, et apporte plusieurs produits dans le contexte actuel.
- Pour remplir l'**id** du côté cible, MapForce suit la règle générale (pour chaque item dans la source, crée un item dans la cible). Néanmoins, comme expliqué ci-dessus, tous les produits provenant de **Products.xml** se trouvent dans le contexte actuel. Il n'y a pas de connexion entre les **product** vers un autre endroit dans la cible afin de lire l'**id** d'un produit spécifique uniquement. En conséquence,

plusieurs éléments **id** seront créés pour chaque **Article** dans la cible. La même chose se produit avec **color** et **size**.

Pour résumer : des items provenant de **Products.xml** possèdent le contexte du filtre (qui doit itérer à travers chaque produit) ; c'est pourquoi, les valeurs **id**, **color** et **size** seront copiées dans chaque cible **row** autant de fois qu'il y a des produits dans le fichier source, et généreront un résultat incorrect comme celui ci-dessous :

```
<rows>
    <row>
        <id>1</id>
        <id>2</id>
        <id>3</id>
        <name>T-Shirt</name>
        <color>red</color>
        <color>blue</color>
        <color>green</color>
        <size>10</size>
        <size>20</size>
        <size>30</size>
        <price>25</price>
    </row>
</rows>
```

Solution A : Utiliser le contexte de priorité

Le problème ci-dessus a été résolu en ajoutant un contexte de priorité dans la fonction qui calcule la condition Booléenne du filtre.

En particulier, si le second paramètres d'entrée de la fonction **equal** est désigné en tant que contexte de priorité, la séquence provenant de **Products.xml** est mis en priorité. Cela se traduit par la logique de mappage suivante :

- Pour chaque produit, remplir l'entrée **b** de la fonction **equal** (en d'autres termes, mettre la priorité sur **b**). À ce niveau, les détails du produit actuel se trouvent dans le contexte.
- Pour chaque article, remplir l'entrée **a** de la fonction **equal** et vérifier si la condition de filtre est vraie. Si oui, placer les détails de l'article également dans le contexte actuel.
- Ensuite, copier les détails de l'article et du produit depuis le contexte actuel dans les items respectifs dans la cible.

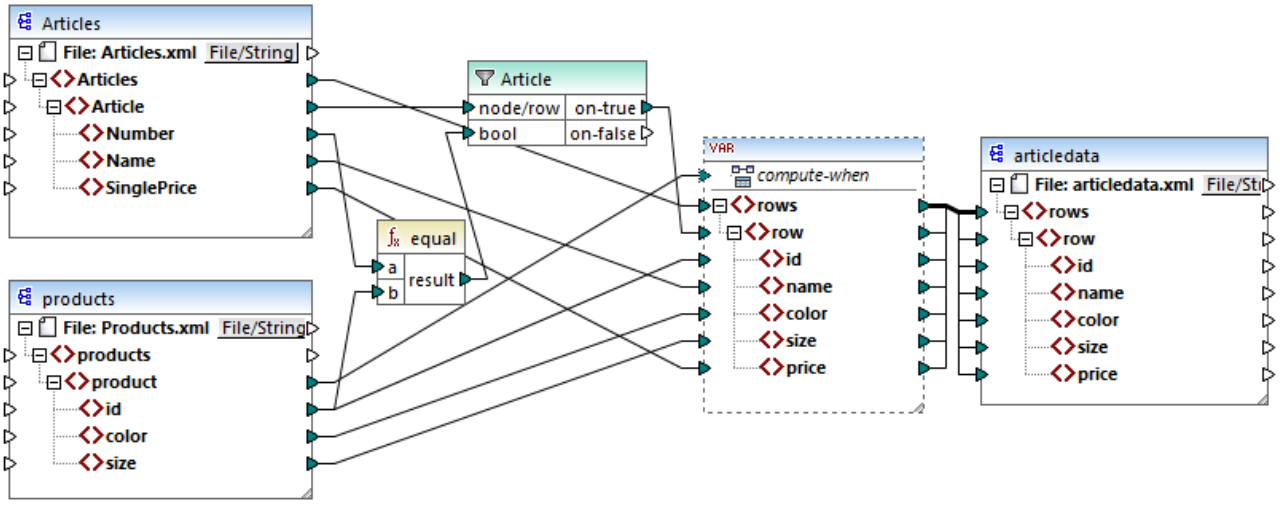
La logique de mappage ci-dessus produit le résultat correct, par exemple :

```
<rows>
    <row>
        <id>1</id>
        <name>T-Shirt</name>
        <color>red</color>
        <size>10</size>
        <price>25</price>
    </row>
</rows>
```

Solution B : Utiliser une variable

En tant qu'une solution alternative, vous pouvez apporter chaque article et produit qui correspond la condition du filtre dans le même contexte avec l'aide d'une variable intermédiaire. Les variables sont pertinentes pour des scénarios comme celui-ci parce qu'elles vous permettent de stocker des données temporairement dans le mappage, et vous aident donc à modifier le contexte selon vos besoins.

Pour des scénarios comme celui-ci, vous pouvez ajouter au mappage une variable qui a le même schéma que le composant de cible. Dans le menu **Insertion**, cliquer sur **Variable**, et fournir le schéma **articledata.xsd** en tant que structure lorsque vous y êtes invité.



Dans le mappage ci-dessus, les choses suivantes se produisent :

- Le contexte de priorité n'est plus utilisé. À la place de cela il y a une variable, qui a la même structure que le composant de cible.
- Comme d'habitude, l'exécution de mappage commence à partir du nœud racine de cible. Avant de remplir la cible, le mappage collectionne des données dans la variable.
- La variable est calculée dans le contexte de chaque produit. Cela se produit parce qu'il y a une connexion provenant de **product** vers l'entrée **compute-when** de la variable.
- La condition de filtre est donc vérifiée dans le contexte de chaque produit. Il faut que la condition soit vraie, la structure de la variable sera remplie et transmise vers la cible.

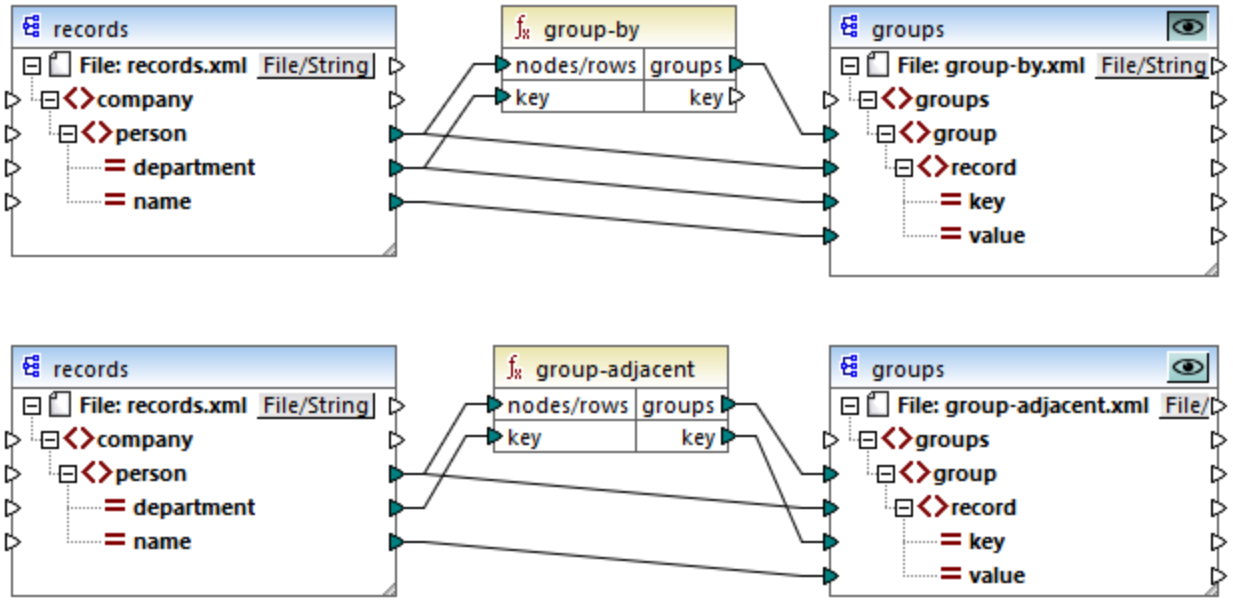
7.3.4 Composants de cible multiple

Un mappage peut avoir plusieurs composants de source et de cible. Lorsqu'il y a plusieurs composants de cible, vous pouvez prévisualiser uniquement une sortie de composant à la fois dans MapForce, celui que vous indiquez en cliquant sur la touche **Preview**. Dans d'autres environnement d'exécution (MapForce Server ou code généré), tous les composants de cible seront exécutés séquentiellement, et la sortie respective de chaque composant sera produite.

Par défaut, les composants de cible sont traités du haut en bas et de gauche à droite. Si nécessaire, vous pouvez influer sur cet ordre en changeant la position des composants de cible dans la fenêtre de mappage. Le point de référence est le coin gauche supérieur de chaque composant. Veuillez noter les points suivants :

- Si deux composants ont la même position verticale, alors celui placé à l'extrême gauche prend précédence.
- Si deux composants ont la même position horizontale, alors celui placé à le plus haut prend précédence.
- Dans la situation improbable que les composants aient exactement la même position, une ID de composant interne unique sera utilisée automatiquement, ce qui garantit un ordre bien défini mais qui ne peut pas être changé.

Pour voir un exemple de son fonctionnement, ouvrir la démo de mappage suivante : **<Documents>\Altova\MapForce2024\MapForceExamples\Tutorial\GroupingFunctions.mfd**. Ce mappage consiste en plusieurs sources et plusieurs composants de cible ; seul un fragment est affiché ci-dessous.



Conformément aux règles, l'ordre de traitement par défaut de ce mappage dans MapForce Server et dans le code généré va du haut en bas. Vous pouvez vérifier que cela est bien le cas en générant un code XSLT 2.0, par exemple.

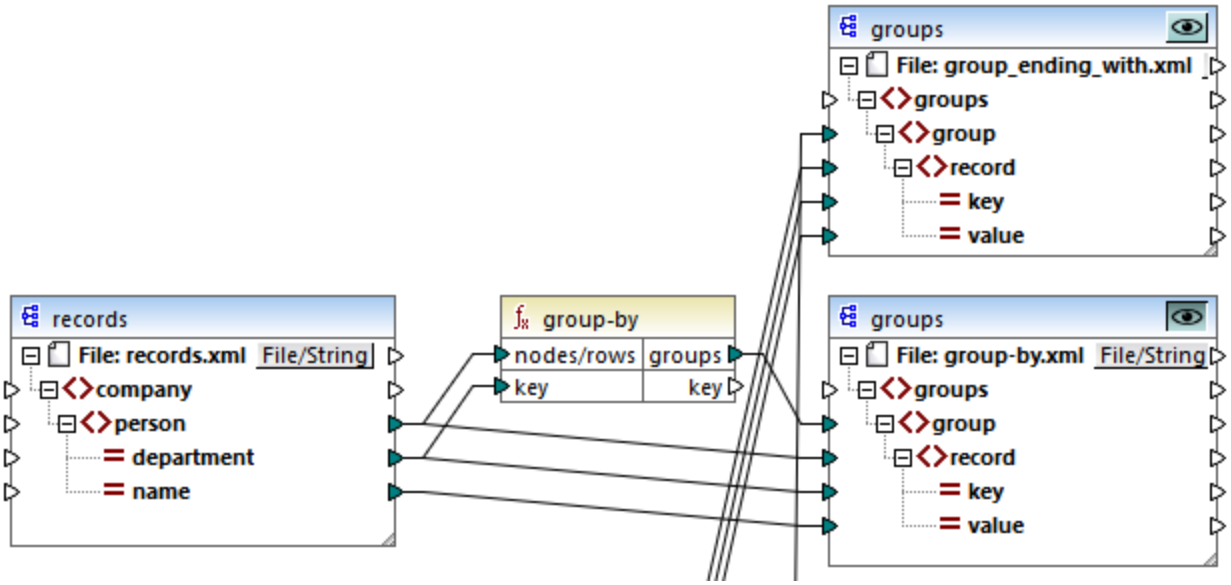
1. Dans le menu **Fichier**, cliquer sur **Générer du code dans | XSLT 2.0**.
2. Lorsque vous êtes invité à le faire, sélectionner un répertoire de cible pour le code généré.

Après la génération, le répertoire de cible inclut plusieurs fichiers XSLT et un fichier **DoTransform.bat**. Ce dernier peut être exécuté par RaptorXML Server (nécessite une licence séparée). Le fichier **DoTransform.bat** traite des composants dans le même ordre qu'ils ont été définis dans le mappage, du haut en bas. Cela peut être vérifié en regardant le paramètre `--output` de chaque transformation.

```
RaptorXML xslt --xslt-version=2 --input="records.xml" --output="group-by.xml" --xml-
validation-error-as-warning=true %* "MappingMapTogroups.xslt"
IF ERRORLEVEL 1 EXIT/B %ERRORLEVEL%
RaptorXML xslt --xslt-version=2 --input="records.xml" --output="group-adjacent.xml" --
xml-validation-error-as-warning=true %* "MappingMapTogroups2.xslt"
IF ERRORLEVEL 1 EXIT/B %ERRORLEVEL%
```

```
RaptorXML xslt --xslt-version=2 --input="records.xml" --output="group-into-blocks.xml" --
xml-validation-error-as-warning=true %* "MappingMapTogroups3.xslt"
IF ERRORLEVEL 1 EXIT/B %ERRORLEVEL%
RaptorXML xslt --xslt-version=2 --input="records-v2.xml" --output="group-starting-
with.xml" --xml-validation-error-as-warning=true %* "MappingMapTogroups4.xslt"
IF ERRORLEVEL 1 EXIT/B %ERRORLEVEL%
RaptorXML xslt --xslt-version=2 --input="records-v3.xml" --output="group_ending_with.xml"
--xml-validation-error-as-warning=true %* "MappingMapTogroups5.xslt"
IF ERRORLEVEL 1 EXIT/B %ERRORLEVEL%
```

La dernière transformation produit un fichier de sortie appelée **group-ending-with.xml**. Déplaçons ce composant de cible sur le mappage tout en haut :



Si vous générez à nouveau le code XSLT 2.0, l'ordre de traitement change en conséquence :

```
RaptorXML xslt --xslt-version=2 --input="records-v3.xml" --output="group_ending_with.xml"
--xml-validation-error-as-warning=true %* "MappingMapTogroups.xslt"
IF ERRORLEVEL 1 EXIT/B %ERRORLEVEL%
RaptorXML xslt --xslt-version=2 --input="records.xml" --output="group-by.xml" --xml-
validation-error-as-warning=true %* "MappingMapTogroups2.xslt"
IF ERRORLEVEL 1 EXIT/B %ERRORLEVEL%
RaptorXML xslt --xslt-version=2 --input="records.xml" --output="group-adjacent.xml" --
xml-validation-error-as-warning=true %* "MappingMapTogroups3.xslt"
IF ERRORLEVEL 1 EXIT/B %ERRORLEVEL%
RaptorXML xslt --xslt-version=2 --input="records.xml" --output="group-into-blocks.xml" --
xml-validation-error-as-warning=true %* "MappingMapTogroups4.xslt"
IF ERRORLEVEL 1 EXIT/B %ERRORLEVEL%
RaptorXML xslt --xslt-version=2 --input="records-v2.xml" --output="group-starting-
with.xml" --xml-validation-error-as-warning=true %* "MappingMapTogroups5.xslt"
IF ERRORLEVEL 1 EXIT/B %ERRORLEVEL%
```

Dans l'extrait de code ci-dessus, le premier appel produit maintenant **group-ending-with.xml**.

Vous pouvez changer l'ordre de traitement d'une manière semblable que dans d'autres langages de code et dans les fichiers d'exécution compilés de MapForceServer (.mfx).

Mappages enchaînés

La même séquence de traitement telle que décrite ci-dessus, est suivie pour les mappages enchaînés. Néanmoins, le groupe de mappage enchaîné est pris en tant qu'une unité. Le repositionnement du composant intermédiaire ou de cible final d'un seul mappage enchaîné n'a pas d'effet sur la séquence de traitement. Ce n'est que si plusieurs "chaînes" ou plusieurs composants de cible existent dans un mappage que la position des composants de cible finaux de chaque groupe détermine lequel est traité en premier.

- Si deux composants finaux de cible ont la même position verticale, alors celui placé à l'extrême gauche prend précédence.
- Si deux composants finaux de cible ont la même position horizontale, alors celui placé à le plus haut prend précédence.
- Dans la situation improbable que les composants aient exactement la même position, une ID de composant interne unique sera utilisée automatiquement, ce qui garantit un ordre bien défini mais qui ne peut pas être changé.

8 Automatisation avec les Produits d'Altova

Les mappages conçus avec MapForce peuvent être exécutés dans un environnement de serveur (y compris les serveurs Linux et macOS) et avec des performances de niveau de serveur, par les moteurs de transformation Altova suivants (mis sous licence séparément) :

- *RaptorXML Server*. L'exécution d'un mappage avec ce moteur est pertinent si le langage de transformation du mappage est XSLT 1.0, XSLT 2.0, XSLT 3.0 ou XQuery. Voir Automatisation avec RaptorXML Server 426.
- *MapForce Server (ou MapForce Server Advanced Edition).* Ce moteur convient pour tout mappage lorsque le langage de transformation est BUILT-IN*. Le langage BUILT-IN prend en charge la plupart des fonctions de mappage dans MapForce, alors que MapForce Server (et en particulier, MapForce Server Advanced Edition) fournit la meilleure performance pour exécuter un mappage.

* *Le langage de transformation BUILT-IN nécessite l'édition MapForce Professional ou Enterprise.*

De plus, MapForce permet d'automatiser la génération de code XSLT depuis l'interface de ligne de commande. Pour plus d'informations, voir Interface de ligne de commande MapForce 427.

8.1 Automatisation avec RaptorXML Server

RaptorXML Server (ci-après abrégé par RaptorXML) est le processeur XML et XBRL hyper-rapide de troisième génération d'Altova. Il est conçu pour être optimisé pour les tous derniers standards et des environnements de calcul parallèles. Conçu pour fonctionner sur plusieurs plateformes, le moteur profite de l'ubiquité actuelle des ordinateurs multicœurs pour fournir un traitement ultra-rapide des données XML et XBRL.

RaptorXML est disponible dans deux éditions qui peuvent être téléchargées sur la page de téléchargement d'Altova (https://www.altova.com/download-trial-server.html):

- RaptorXML Server est un moteur de traitement XML très rapide qui prend en charge XML, Schéma XML, XSLT, XPath, XQuery, etc.
- RaptorXML+XBRL Server prend en charge toutes les fonctions de RaptorXML Server avec en outre des fonctions permettant de traiter et de valider la famille XBRL de standards.

Si vous générez du code dans XSLT, MapForce crée un fichier batch appelé **DoTransform.bat** qui est placé dans le dossier de sortie que vous choisissez lors de la génération. L'exécution du fichier batch appelle RaptorXML Server et exécute la transformation XSLT sur le serveur.
Note : Vous pouvez aussi consulter le code XSLT[65] à l'aide du moteur intégré.

8.2 Interface de ligne de commande MapForce

La syntaxe générale d'une commande MapForce sous la ligne de commande :

```
MapForce.exe <filename> [/{target} [[<outputdir>] [/options]]]
```

Pour plus d'information sur chaque paramètre de la commande, voir la liste ci-dessous.

Syntaxe de ligne de commande
Les annotations suivantes sont utilisées pour indiquer une syntaxe de ligne de commande :

Notation	Description
Texte sans crochets ou parenthèses	Items que vous devez saisir comme indiqué
`<Texte dans des crochets pointus>`	Caractère générique pour laquelle vous devez fournir une valeur
`[Texte dans des crochets]`	Items optionnels
`{Texte dans des accolades}`	Ensemble d'items requis ; en choisir un
Barre verticale `(|)`	Le séparateur pour des items mutuellement exclusifs ; en choisir un
Ellipse `(...)`	Items pouvant être répétés

`<filename>`

Le fichier de design de mappage (**.mfd**) ou le projet de mappage (**.mfp**) (*éditions Professional et Enterprise*) à partir duquel le code doit être généré.

`/{target}`

Spécifie le langage cible ou l'environnement pour lequel le code doit être généré. Les cibles de génération de code suivantes sont prises en charge.

- `/XSLT`
- `/XSLT2`
- `/XSLT3`

`<outputdir>`

Paramètre optionnel qui spécifie le répertoire de sortie. Si un chemin de sortie n'est pas fourni, le répertoire de travail actuel sera utilisé. Veuillez noter que tout chemin de fichier relatif sera relatif par rapport au répertoire de travail actuel.

`/options`

Les `/options` ne sont pas exclusives mutuellement. Une ou plusieurs des options suivantes peuvent être définies :

- L'option `/GLOBALRESOURCEFILE <filename>` est applicable si le mappage utilise des Ressources globales pour résoudre un fichier d'entrée ou de sortie ou des chemins de dossier ou des bases

de données. Pour plus d'informations, voir Ressources globales Altova[430].
L'option `/GLOBALRESOURCEFILE` spécifie le chemin vers un fichier XML de Ressources globales. Veuillez noter que, si `/GLOBALRESOURCEFILE` est défini, alors `/GLOBALRESOURCECONFIG` doit aussi être défini.

- L'option `/GLOBALRESOURCECONFIG <config>` spécifie le nom de la configuration de Ressource globale (*voir aussi la version précédente*). Veuillez noter que, si `/GLOBALRESOURCEFILE` est défini, alors `/GLOBALRESOURCECONFIG` doit aussi être défini.

- L'option `/LOG <logfilename>` génère un fichier log dans le chemin spécifié. Le chemin `<logfilename>` peut être un chemin absolu. Si un chemin complet est fourni, le répertoire doit exister pour que le fichier de journal soit généré. Si vous spécifiez uniquement le nom de fichier, celui-ci sera placé dans le répertoire actuel de l'invite de commande Windows.

Notes

- Les chemins relatifs sont relatifs par rapport au répertoire de travail, qui est le répertoire actuel de l'application appelant MapForce. Cela s'applique au chemin du nom de fichier **`.mfd`**, au répertoire de sortie, au nom de fichier de journal, et au nom de fichier de ressource globale.
- Ne pas utiliser la barre oblique de la fin et les guillemets de fermeture au niveau de la ligne de commande (par exemple, **`"C:\My directory\"`**). Ces deux caractères sont interprétés par le parseur de ligne de commande en tant que double guillemet littéral. Il est recommandé d'éviter les espaces et les guillemets. Si les espaces apparaissent dans la ligne de commande et si vous avez besoin de guillemets, utilisez les barres obliques inverses doubles (par ex., **`"c:\My Directory\\"`**).

Exemples

1) Pour lancer MapForce et ouvrir le mappage `<filename>.mfd`, utilisez la commande suivante :

```
MapForce.exe <filename>.mfd
```

2) Pour générer un code XSLT 2.0 et créer un fichier log portant le nom `<logfilename>`, utilisez la commande suivante :

```
MapForce.exe <filename>.mfd /XSLT2 <outputdir> /LOG <logfilename>
```

3) Pour générer un code XSLT 2.0 en prenant en compte la configuration de ressource globale `<grconfigname>` depuis le fichier de ressource globale `<grfilename>`, utiliser :

```
Mapforce.exe <filename>.mfd /XSLT2 <outputdir> /GLOBALRESOURCEFILE
<grfilename> /GLOBALRESOURCECONFIG <grconfigname>
```

Exemple d'éditions Professional et Enterprise

1) Pour générer une application C# pour Visual Studio 2022 et sortir un fichier log, utilisez la commande suivante :

```
MapForce.exe <filename>.mfd /CS:VS2022 <outputdir> /LOG <logfilename>
```

2) Pour générer une application C++ en utilisant les paramètres de génération de code définis dans **Outils | Options**, et sortir un fichier log, utiliser :

```
MapForce.exe <filename>.mfd /CPP <outputdir> /LOG <logfilename>
```

3) Pour générer une application C++ pour Visual Studio 2022, MSXML, avec des bibliothèques statiques, prise en charge de MFC et pas de fichier log, utilisez la commande suivante :

```
MapForce.exe <filename>.mfd /CPP:VS2022,MSXML,LIB,MFC
```

4) Pour générer une application C++ pour Visual Studio 2022, Xerces, avec des bibliothèques dynamiques, pas de prise en charge MFC et un fichier log, utilisez la commande suivante :

```
MapForce.exe <filename>.mfd /CPP:VS2022,XERCES,DLL,NoMFC <outputdir> /LOG
<logfilename>
```

5) Pour générer une application Java et sortir un fichier log, utilisez la commande suivante :

```
MapForce.exe <filename>.mfd /JAVA <outputdir> /LOG <logfilename>
```

6) Pour générer un code pour tous les mappages dans le projet, en utilisant le langage et le répertoire de sortie définis dans les paramètres du dossier (de chaque dossier dans le projet), utilisez la commande suivante :

```
MapForce.exe <filename>.mfp /GENERATE /LOG <logfilename>
```

7) Pour générer un code Java pour tous les mappages dans le fichier de projet, utilisez la commande suivante :

```
MapForce.exe <filename>.mfp /JAVA /LOG <logfilename>
```

Veuillez noter que le langage de génération de code défini dans les paramètres de dossier sont ignorés et que Java est utilisé pour tous les mappages.

8) Pour fournir des fichiers d'entrée et de sortie dans la ligne de commande pour un mappage Java compilé précédemment, utilisez les commandes suivantes :

```
java -jar <mappingfile>.jar /InputFileName <inputfilename> /OutputFileName
<outputfilename>
```

Les paramètres `/InputFileName` et `/OutputFileName` sont les noms de composants d'entrée spéciaux dans le mappage MapForce qui vous permettent d'utiliser des paramètres dans l'exécution de la ligne de commande (voir Fournir des paramètres au mappage 147).

9) Pour compiler un mappage dans un fichier d'exécution MapForce Server, pour la version MapForce Server 2024, et réprimant les signatures XML :

```
MapForce.exe <filename>.mfd /COMPILE:NOXMLSIGNATURES
<outputdir> /MFXVERSION:2024 /LOG <logfilename>
```

9 Ressources globales Altova

Les Ressources globales d'Altova sont des alias pour des fichiers, dossiers et ressources de bases de données. Chaque alias peut avoir de multiples configurations et chaque configuration correspond à une seule ressource. Pour cette raison, quand vous utilisez une ressource globale, vous pouvez basculer entre ses configurations. Par exemple, vous pouvez créer une ressource "database" avec deux configurations : `développement` et `production`. Dépendant de vos objectifs, vous pouvez basculer entre ces configurations.

Les Ressources globales peuvent être utilisées dans toutes les différentes applications d'Altova (*voir la sous-section ci-dessous*).

Ressources globales dans d'autres produits d'Altova

Lorsqu'ils sont stockés en tant que Ressources globales, les détails de connexion à la base de données deviennent réutilisables et disponibles dans plusieurs applications d'Altova. Par exemple, si vous avez souvent besoin d'ouvrir le même fichier dans plusieurs applications desktop Altova, vous pourriez le définir en tant que Ressources globales. Si vous devez modifier le chemin de fichier, vous allez devoir le changer à un endroit uniquement. Actuellement, les Ressources globales peuvent être définies et utilisées dans les produits Altova suivants :

- Altova Authentic
- DatabaseSpy
- MobileTogether Designer
- MapForce
- StyleVision
- XMLSpy
- FlowForce Server
- MapForce Server
- RaptorXML Server et RaptorXML+XBRL Server.

Dans cette section

Cette section explique comment créer et configurer différents types de ressources globales. La section est organisée en rubriques suivantes :

9.1 Configuration des Ressources globales, Partie 1

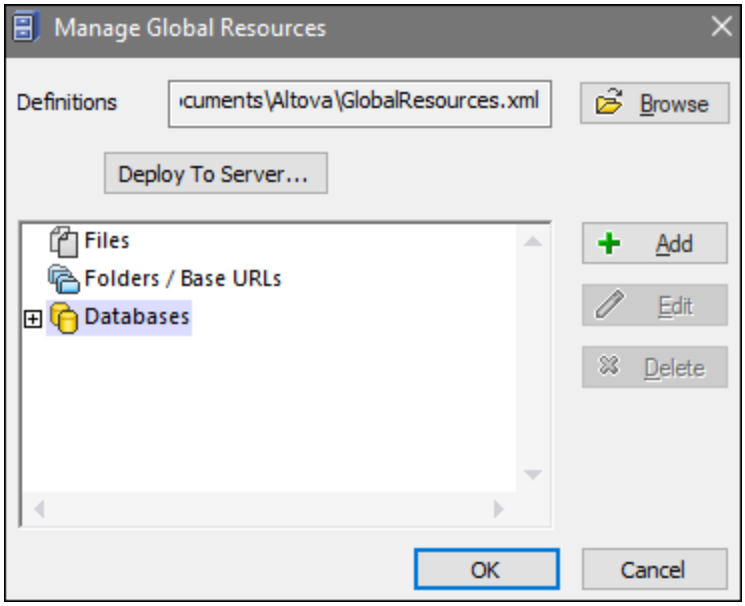
La configuration des Ressources globales a lieu en deux parties : (i) créer une ressource globale dans la boîte de dialogue **Gérer les Ressources globales** (*voir ci-dessous*) et (ii) définir les propriétés de cette ressource globale dans le dialogue **Ressource globale**. La deuxième partie est discutée dans la prochaine rubrique 433.

Les Ressources globales d'Altova sont définies dans le dialogue **Gérer les Ressources globales**, qui peuvent être accédées de deux manières :

- Cliquez sur la commande de menu **Outils | Ressources globales**.
- Cliquez sur l'icône **Gérer les Ressources globales** dans la barre d'outils des Ressources globales (*voir la capture d'écran ci-dessous*).

Fichier de définition des ressources globales.

Les informations sur les ressources globales sont stockées dans le fichier XML appelé fichier de définition des Ressources globales. Ce fichier est créé lorsque la première ressource globale est définie dans la boîte de dialogue **Gérer les Ressources globales** (*voir la capture d'écran ci-dessous*) et enregistré.



Lorsque vous ouvrez la boîte de dialogue **Gérer les Ressources globales** pour la première fois, l'emplacement par défaut et le nom du fichier de définition des Ressources globales sont spécifiés dans la zone de texte *Définitions* (*voir la capture d'écran ci-dessus*):

```
C:\Users\<username>\Documents\Altova\GlobalResources.xml
```

Ce fichier est défini comme fichier de définition des Ressources globales par défaut pour toutes les applications d'Altova. Une ressource globale peut être enregistrée depuis toute application d'Altova dans ce fichier et sera immédiatement disponible dans toutes les autres applications en tant que ressource globale. Pour définir et

enregistrer une ressource globale dans le fichier de définition des ressources globales, ajoutez une ressource globale dans le dialogue **Gérer les Ressources globales** et cliquez sur **OK** pour l'enregistrer.

Pour sélectionner un fichier de définition des ressources globales déjà existant afin d'en faire le fichier de définition des ressources globales d'une application d'Altova spécifique, recherchez-le par le biais du bouton **Naviguer** de la zone de texte *Définitions* (*voir la capture d'écran ci-dessus*).

La boîte de dialogue **Gérer les Ressources globales** vous permet également d'éditer et de supprimer des ressources globales existantes.

Notes :

- Vous pouvez donner n'importe quel nom au fichier de définition des ressources globales et l'enregistrer à un emplacement accessible à vos applications Altova. Tout ce que vous devez faire dans chaque application est spécifier ce fichier comme fichier de définition des Ressources globales pour cette application (dans la zone de texte *Définitions*). Les ressources deviennent des produits à échelle globale d'Altova lorsque vous utilisez un seul fichier de définitions dans l'ensemble des produits globaux d'Altova.

- Vous pouvez aussi créer de multiples fichiers de définition des ressources globales. Toutefois, seulement un de ces fichiers peut être actif à tout moment dans une application donnée d'Altova, et uniquement les définitions contenues dans ce fichier seront disponibles pour l'application. La disponibilité des ressources peut être restreinte pour cette raison ou conçue pour se chevaucher sur l'ensemble des produits, tel que requis.

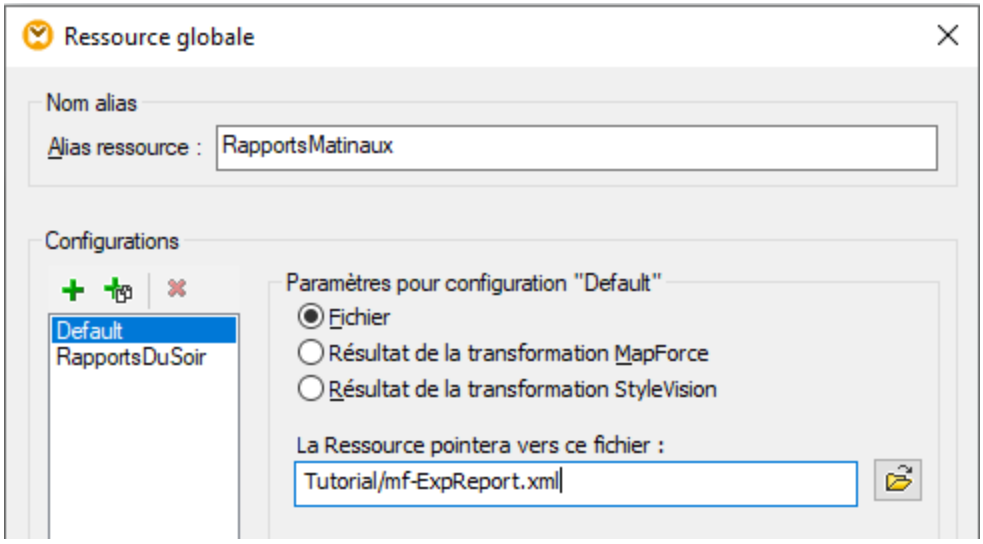
9.2 Configuration des Ressources globales, Partie 2

La deuxième partie de la configuration des ressources globales consiste en la définition de propriétés d'une ressource globale dans la boîte de dialogue **Ressource globale**. Les propriétés dépendent du type de ressource globale (*voir les sous-sections ci-dessous*). Vous pouvez accéder la boîte de dialogue **Ressource globale** en cliquant sur le bouton **Ajouter** dans la boîte de dialogue Gérer les Ressources globales 431.

Pour en savoir plus sur la configuration des différents types de ressources globales, voir les exemples suivants : Fichiers XML en tant que Ressources globales 436, Dossiers en tant que Ressources globales 438.

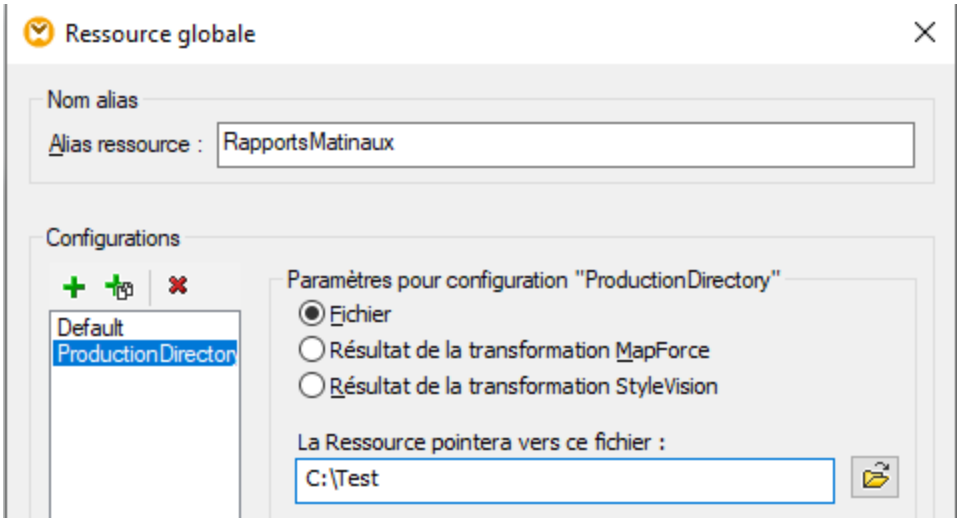
Fichiers

Les propriétés spécifiques au fichier sont affichées dans la boîte de dialogue **Ressource globale** ci-dessous. La configuration est divisée en trois parties majeures : (i) le nom du fichier, (ii) l'emplacement de ce fichier, et (iii) la liste de configurations définies dans l'alias du fichier.



Dossiers

Les propriétés spécifiques au dossier sont affichées dans la boîte de dialogue **Ressource globale** ci-dessous. La configuration est divisée en trois parties majeures : (i) le nom du dossier, (ii) l'emplacement de ce dossier, et (iii) la liste de configurations définies dans l'alias du fichier.



Icônes du dialogue Ressource globale

	Ajouter une configuration : Affiche la boîte de dialogue Ajouter une configuration dans laquelle vous saisissez le nom de la configuration à ajouter.
	Ajouter une configuration en tant que copie : Affiche la boîte de dialogue Ajouter une configuration dans laquelle vous pouvez saisir le nom de la configuration à créer en tant que une copie de la configuration choisie.
	Supprimer : Supprime la configuration sélectionnée.
	Ouvrir : Rechercher le fichier à créer en tant que ressource globale.
	Ouvrir : Rechercher le dossier à créer en tant que ressource globale.

Configuration de la Ressources globale : Procédures générales

La procédure générale de création et de configuration de ressources globales est décrite ci-dessous :

1. Cliquez sur le bouton de la barre d'outils (**Gérer les Ressources globales**). En alternative, allez au menu **Outils** and cliquez sur **Ressources globales**.
2. Cliquez sur **Ajouter** et choisissez le type de ressource que vous souhaitez créer (fichier, dossier, base de données). La boîte de dialogue **Ressource globale** apparaîtra.
3. Saisir un nom descriptif dans la zone de texte **Alias de ressource** (par ex.,`InputFile`).

4. Définir la configuration par défaut dépend du type de la ressource globale : (i) pour un fichier ou dossier, recherchez le fichier ou dossier vers lequel cette ressource devrait pointer par défaut ; (ii) pour une connexion de base de données, cliquez sur **Choisir base de données** et suivez l'assistant de connexion à la base de données pour se connecter à la base de données . Cette connexion de base de données sera utilisée par défaut lorsque vous exécutez le mappage.
5. Si vous avez besoin d'une configuration supplémentaire (par ex., un dossier de sortie additionnel), cliquez sur le bouton dans la boîte de dialogue **Ressource globale**, saisissez le nom de cette configuration et spécifiez le chemin vers cette configuration.
6. Répétez l'étape précédente pour chaque configuration supplémentaire requise.

Note : les connexions de base de données sont prises en charge comme ressources globales uniquement dans les éditions de MapForce Professional and Enterprise.

9.3 Fichiers XML en tant que Ressources globales

Cette rubrique explique comment utiliser les fichiers XML en tant que ressources globales. Il existe des situations où vous allez éventuellement devoir changer le fichier XML de nombreuses fois par jour. Par exemple, chaque matin, vous devez exécuter un mappage particulier et générer un rapport en utilisant un fichier XML en tant qu'entrée de mappage, et chaque soir le même rapport doit être généré depuis un autre fichier XML. Au lieu d'éditer le mappage plusieurs fois par jour (ou garder de multiples copies de celui-ci), vous pourriez configurer le mappage de telle manière qu'il lise d'un fichier défini comme ressource globale (un genre de *alias du fichier*). Dans cet exemple, vos alias de fichier auront deux configurations :

1. La configuration `par défaut` fournit un fichier XML « matinal » en tant qu'entrée de mappage.
2. La configuration `RapportsDuSoir` fournit un fichier XML « du soir » en tant qu'entrée de mappage.

Pour créer et configurer un alias de fichier, suivez les étapes ci-dessous.

Étape 1 : Créer une ressource globale

D'abord, nous devons créer un alias de fichier. Suivez les instructions ci-dessous :

1. Cliquez sur le bouton de la barre d'outils (**Gérer les Ressources globales**). En alternative, allez au menu **Outils** and cliquez sur **Ressources globales**.
2. Cliquez sur **Ajouter | Fichier** et saisissez le nom dans la zone de texte **alias de Ressource**. Dans cet exemple, nous appelons notre configuration par défaut `RapportsMatinaux`.
3. Cliquez sur le bouton **Parcourir** à côté du champ de texte **La Ressource pointera vers ce fichier** et sélectionnez **`Tutorial\mf-ExpReport.xml`**.
4. Cliquez sur dans la section **Configurations** et nommez cette deuxième configuration `RapportsDuSoir`.
5. Cliquez sur **Parcourir** et sélectionnez **`Tutorial\mf-ExpReport2.xml`**.

Étape 2 : Utiliser la Ressource Globale dans le mappage

Maintenant, nous pouvons utilisez la ressource globale nouvellement créée dans notre mappage. Pour que le mappage lise de la ressource globale, suivez les étapes ci-dessous :

1. Ouvrez le mappage **`Tutorial\Tut-ExpReport.mfd`**.
2. Double-cliquez sur l'en-tête du composant source pour ouvrir une boîte de dialogue **Paramètres de composant**.
3. À côté du **fichier XML d'entrée**, cliquez sur **Parcourir**, puis cliquez sur **Ressources globales** et sélectionnez l'alias de fichier `RapportsMatinaux`. Cliquer sur **Ouvrir**.
4. Ouvrez une nouvelle fois la boîte de dialogue **Paramètres de composant** : Le chemin de fichier XML d'entrée est maintenant devenu **`altova://file_resource/MorningReports`**, ce qui indique que le chemin utilise une Ressource Globale.

Étape 3 : Exécuter le mappage avec la configuration désirée

Vous pouvez désormais basculer entre les fichiers XML d'entrée avant d'exécuter le mappage :

- Pour utiliser **`mf-ExpReport.xml`** comme entrée, sélectionnez l'élément de menu **Outils | Configuration active | Par défaut**.
- Pour utiliser **`mf-ExpReport2.xml`** comme entrée, sélectionnez l'item de menu **Outils | Configuration active | RapportsDuSoir**.

En alternative, sélectionnez la configuration requise depuis la liste déroulante **Ressources Globales** dans la barre d'outils (*voir la capture d'écran ci-dessous*).

Pour consulter le résultat de mappage directement dans MapForce, cliquez sur l'onglet **Sortie**.

9.4 Dossiers en tant que Ressources globales

Cette rubrique explique comment utiliser les dossiers en tant que ressources globales. Il existe des situations dans lesquelles vous devrez générer la même sortie dans différents répertoires. À cette fin, nous devons créer un alias de dossier avec deux configurations :

1. La configuration `Default` générera la sortie dans **`C:\Test`**.
2. La configuration `Production` générera la sortie dans **`C:\ par`**.

Pour créer et configurer un alias de dossier, suivez les étapes ci-dessous.

Étape 1 : Créer une ressource globale

D'abord, nous devons créer un alias de dossier. Suivez les instructions ci-dessous :

1. Cliquez sur le  bouton de la barre d'outils (**Gérer les Ressources globales**). En alternative, allez au menu **Outils** and cliquez sur **Ressources globales**.
2. Cliquez sur **Ajouter | Dossier** et saisissez un nom dans la zone de texte **alias de Ressource**. Dans cet exemple, nous appelons notre configuration par défaut `OutputDirectory`.
3. Cliquez sur le bouton **Parcourir** à côté du champ de texte **Paramètres pour une configuration "Par défaut"** et sélectionnez **`C:\Test`**. Assurez-vous que ce dossier existe déjà dans votre système d'exploitation.
4. Cliquez sur  et saisissez un nom pour la deuxième configuration. Dans cet exemple, nous appelons notre deuxième configuration `ProductionDirectory`.
5. Cliquez sur **Parcourir** et sélectionnez le dossier **`C:\Production`**. Assurez-vous que ce dossier existe déjà dans votre système d'exploitation.

Étape 2 : Utiliser la Ressource Globale dans le mappage

La prochaine étape est de faire en sorte que le mappage utilise l'alias de dossier que nous venons de créé. Suivez les étapes ci-dessous :

1. Ouvrez le mappage **`Tutorial\Tut-ExpReport.mfd`**.
2. Double-cliquez sur l'en-tête du composant cible pour ouvrir la boîte de dialogue **Paramètres de composant**.
3. Cliquez sur **Ressources globales**, puis cliquez sur **Enregistrer**.
4. Enregistrez le fichier XML de sortie comme **`Output.xml`**. Le chemin de fichier XML de sortie est maintenant devenu **`altova://folder_resource/OutputDirectory/Output.xml`**, qui indique que le chemin est défini comme ressource globale.

Étape 3 : Exécuter le mappage avec la configuration désirée

Vous pouvez désormais basculer entre les dossiers de sortie avant d'exécuter le mappage :

- Pour utiliser **`C:\Test`** en tant que configuration de sortie, sélectionnez l'item de menu **Outils | Configuration active | Par défaut**.
- Pour utiliser **`C:\Production`** comme répertoire de sortie, sélectionnez l'item de menu **Outils | Configuration active | ProductionDirectory**.

Par défaut, la sortie de mappage est écrite en tant que fichier temporaire, à moins que vous ayez configuré explicitement MapForce pour écrire les sorties vers les fichiers permanents. Pour configurer MapForce afin qu'il génère les fichiers générés, suivez les étapes suivantes :

1. Allez au menu **Outils** et cliquez sur **Options**.
2. Dans la section **Généralités**, choisissez l'option **Écrire directement dans les fichiers de sortie finaux**.

10 Catalogs in MapForce

MapForce prend en charge un sous-ensemble de mécanismes de catalogue XML OASIS. Le mécanisme du catalogue permet à MapForce d'extraire des schémas communément utilisés (y compris d'autres fichiers) des dossiers utilisateurs locaux. Cela augmente la vitesse de traitement générale, permet aux utilisateurs de travailler hors ligne (c'est-à-dire sans connexion à un réseau) et améliore la portabilité des documents (parce que les URI ne devraient être modifiés uniquement dans les fichiers catalogue.)

Le mécanisme du catalogue dans MapForce fonctionne comme suit dans la présente section :

Pour plus d'informations sur les catalogues, voir la spécification de catalogues XML.

10.1 Comment fonctionnent les catalogues

Les catalogues peuvent être utilisés pour rediriger les Schémas DTD et XML. Alors que le concept derrière les mécanismes dans les deux cas est le même, les détails sont différents et expliqués ci-dessous.

DTD

Les catalogues sont communément utilisés pour rediriger un appel vers un DTD ou un URI local. Pour ce faire, des identifiants publics ou système sont mappés dans le fichier catalogue vers l'URI local requis. Donc, si la déclaration **DOCTYPE** dans un fichier XML est lue, son identifiant public ou système localise la ressource locale requise par le biais du mappage du fichier catalogue.

Pour les schémas populaires, l'identifiant **PUBLIC** est normalement prédéfini, requérant uniquement que l'URI dans le fichier catalogue mappe l'identifiant **PUBLIC** à la copie locale correcte. Lorsque le document XML est parsé, l'identifiant **PUBLIC** qui le compose est lu. Si cet identifiant est trouvé dans un fichier catalogue, l'URL correspondant dans le catalogue fichier sera consulté et le schéma sera lu depuis cet emplacement. Donc, par exemple, si le fichier SVG est ouvert dans 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>
```

L'identifiant **PUBLIC** de ce fichier SVG file est recherché dans le catalogue. Disons que le fichier catalogue contient l'entrée suivante :

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

Dans ce cas, il y a une correspondance pour l'identifiant **PUBLIC**. En conséquence, le lookup pour SVG DTD est redirigé vers l'URL **schemas/svg/svg11.dtd** (qui est associé au fichier catalogue). Il s'agit d'un fichier local qui sera utilisé en tant que DTD pour le fichier SVG. S'il n'y a pas de mappage pour l'ID **Public** dans le catalogue, l'URL dans le document XML sera utilisé (dans l'exemple du fichier SVG ci-dessus, l'URL Internet est la suivante : **http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd**).

Schémas XML

Dans MapForce, vous pouvez utiliser des catalogues avec des **Schémas XML**. Dans le fichier d'instance XML, la référence au schéma apparaîtra dans l'attribut **xsi:schemaLocation** de l'élément de premier niveau du document XML. Par exemple,

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

La valeur de l'attribut **xsi:schemaLocation** a deux parties : une partie d'espace de noms (vert ci-dessus) et une partie URI (en surbrillance). La partie d'espace de noms est utilisée dans le catalogue pour effectuer le mappage vers la ressource alternative. Par exemple, la saisie catalogue suivante redirige la référence du schéma ci-dessus vers un schéma à un emplacement alternatif.

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

Normalement, la partie URI de la valeur de l'attribut **xsi:schemaLocation** est le chemin vers l'emplacement actuel du schéma. Toutefois, si le schéma est référencé par le biais du catalogue, la partie URI doit pointer vers le schéma XML actuel mais doit exister pour que la validité lexicale de l'attribut **xsi:schemaLocation** soit maintenu. Une valeur `foo`, par exemple, suffirait à la partie URI de la valeur de l'attribut pour qu'elle soit valide.

10.2 Structure du catalogue dans MapForce

Lorsque MapForce est lancé, il charge un fichier désigné **RootCatalog.xml** (*la structure est affichée dans la liste ci-dessous*), qui contient une liste des fichiers catalogue qui seront consultés. Vous pouvez modifier ce fichier et saisir autant de fichiers catalogue que vous souhaitez consulter, chacun est référencé dans un élément **nextCatalog**. Ces fichiers catalogue sont consultés et les URI contenus sont résolus conformément à leurs mappages.

Liste du RootCatalog.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<catalog xmlns="urn:oasis:names:tc:entity:xmlns:xml:catalog"
   xmlns:spy="http://www.altova.com/catalog_ext"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="urn:oasis:names:tc:entity:xmlns:xml:catalog Catalog.xsd">
  <nextCatalog catalog="%PersonalFolder%/Altova/%AppAndVersionName%/CustomCatalog.xml"/>
  <!-- Inclut tous les catalogues sous les schémas communs du premier répertoire dans
l'arborescence -->
  <nextCatalog spy:recurseFrom="%CommonSchemasFolder%" catalog="catalog.xml"
spy:depth="1"/>
  <nextCatalog spy:recurseFrom="%ApplicationWritableDataFolder%/pkgs/.cache"
catalog="remapping.xml" spy:depth="0"/>
  <nextCatalog catalog="CoreCatalog.xml"/>
</catalog>
```

La liste des références ci-dessus renvoie à un catalogue personnalisé (désigné **CustomCatalog.xml**) et à un ensemble de catalogues qui localisent des schémas communément utilisés (tels que les Schémas W3C XML et le schéma SVG).

- **CustomCatalog.xml** est situé dans le sous-dossier De votre dossier personnel (situé par le biais de la variable **%PersonalFolder%**). CustomCatalog.xml est fichier squelette dans lequel vous pouvez créer vos propres mappages. Vous pouvez ajouter des mappages au **CustomCatalog.xml** pour chaque schéma dont vous avez besoin qui n'est pas adressé par les fichiers catalogue dans le dossier des Schémas communs. Pour ce faire, utilisez les éléments pris en charge par le mécanisme de catalogue OASIS (*voir prochaine section*).
- Le dossier des Schémas communs (localisé par la variable **%CommonSchemasFolder%**) contient un ensemble de schémas communément utilisés. À l'intérieur de chaque dossier de schéma, il y a un fichier **catalog.xml** qui mappe des identifiants publics et/ou de système aux URI qui dirigent vers des copies enregistrées localement des schémas respectifs.
- **CoreCatalog.xml** est situé dans le dossier d'application in the MapForce et est utilisé pour localiser les schémas et les feuilles de style utilisés par des processus spécifiques de MapForce, tels que StyleVision Power Stylesheets qui sont des feuilles de style utilisées pour générer le mode Authentique des documents XML de Altova.

Variables d'emplacement

Les variables utilisées dans **RootCatalog.xml** (*liste ci-dessus*) ont les valeurs suivantes :

%PersonalFolder%	Dossier personnel de l'utilisateur actuel, par exemple C:\Users\<name>\Documents
%CommonSchemasFolder%	C:\ProgramData\Altova\Common2024\Schemas

% ApplicationWritableDataFolde r%	C:\ProgramData\Altova

Emplacement des fichiers catalogue et des schémas
Veuillez noter l'emplacement des différents fichiers catalogue.

- **RootCatalog.xml** et **CoreCatalog.xml** sont dans le dossier d'application MapForce.
- **CustomCatalog.xml** est situé dans votre dossier **MyDocuments\Altova\MapForce**.
- Les fichiers **catalog.xml** sont chacun dans un dossier de schéma spécifique, ces dossiers de schéma étant dans le dossier des Schémas communs.

10.3 Personnaliser vos catalogues

Lorsque vous créez des entrées dans **CustomCatalog.xml** (ou tout autre fichier catalogue qui doit être lu par MapForce), utilisez uniquement les éléments suivants de la spécification de catalogue OASIS. Chacun des éléments ci-dessous est répertorié avec une explication de leurs valeurs attribut. Pour plus d'informations sur les catalogues, voir la spécification des catalogues XML. Notez que chaque élément peut prendre l'attribut **xml:base**, qui est utilisé pour spécifier la base URI de cet élément.

- `<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"/>`

Veuillez noter les points suivants :

- Dans le cas où il n'y a pas d'identifiant public, comme pour toutes les feuilles de style, l'identifiant système peut être directement mappé avec un URL par le biais de l'élément **système**.
- Un URI peut être mappé avec un autre URI en utilisant l'élément `uri`.
- Les éléments `rewriteURI` et `rewriteSystem` permettent la réécriture respectivement de la partie initiale d'un URI ou d'un identifiant système. Ceci permet de lancer un chemin de fichier à remplacer et, par conséquent, permet de cibler un autre répertoire. Pour plus d'informations sur les éléments, voir la spécification des catalogues XML.

À partir de la version 2014, MapForce adhère étroitement à la spécification spécification des catalogues XML (OASIS Standard V1.1, 7 octobre 2005). Cette spécification sépare strictement les look-up d'identifiants externes (ceux avec une ID Publique ou une ID Système) des look-up URI (les URI qui sont pas des ID Publiques ou des ID Système). Les URI d'espace de noms doivent donc être considérés comme étant des URI simples —et pas des ID Publiques ou des ID Système—et doivent être utilisés en tant que look-up URI plutôt que des look-up d'identifiants externes. Dans les versions MapForce antérieures à la version 2014, les URI d'espace de noms ont été traduits par les mappages **<public>**. À partir de la version 2014, les mappages **<uri>** doivent être utilisés.

Avant v2014 : `<public publicID="http://www.MyMapping.com/ref" uri="file:///C:/MyDocs/Catalog/test.xsd"/>`
à partir de V-2014 : `<uri name="http://www.MyMapping.com/ref" uri="file:///C:/MyDocs/Catalog/test.xsd"/>`

Comment MapForce trouve un schéma référencé

Un schéma est référencé dans un document XML par le biais de l'attribut **xsi:scemaLocation** (*voir ci-dessous*). La valeur de l'attribut **xsi:schemaLocation** a deux parties : une partie d'espace de noms (vert) et une partie URI (en surbrillance).

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

Ci-dessous, vous trouverez les étapes à suivre pour trouver un schéma référencé, suivies de manière séquentielle par MapForce. Le schéma est chargé lors de la première étape réussie.

1. Consultez le catalogue pour la partie URI de la valeur **xsi:schemaLocation**. Si un mappage est trouvé, y compris dans les mappages **rewriteURI**, utilisez l'URI qui en résulte pour charger le schéma.
2. Consultez le catalogue pour la partie espace de noms de la valeur **xsi:schemaLocation**. Si un mappage est trouvé, y compris dans les mappages **rewriteURI**, utilisez l'URI qui en résulte pour charger le schéma.
3. Utilisez la partie de l'URI de la valeur **xsi:schemaLocation** pour charger le schéma.

Spécifications de schéma XML

L'information de spécification de schéma XML est prédéfinie dans MapForce et la validité des documents de schéma XML (`.xsd`) est comparée à l'information interne. Pour cela, dans un document de schéma XML, il ne devrait pas y avoir de références faites à n'importe quel schéma qui définit la spécification de schéma XML.

Le fichier **catalog.xml** dans le dossier `%AltovaCommonSchemasFolder%\Schemas\schema` contient des références aux DTD qui implémentent des spécifications de schéma XML antérieures. Vous ne devriez pas valider vos documents de schéma XML par rapport à ces schémas. Les fichiers référencés sont inclus uniquement pour donner à MapForce des informations sur les assistants de saisie à des fins d'édition si vous vouliez créer des documents conformément à ces recommandations antérieures.

10.4 Variables d'Environnement

Les variables d'environnement shell peuvent être utilisées dans l'élément `nextCatalog` pour spécifier le chemin menant aux différents emplacements de système (*voir RootCatalog.xml liste ci-dessus*). Les variables d'environnement shell suivantes sont prises en charge :

`%PersonalFolder%`	Chemin complet vers le dossier Personnel de l'utilisateur actuel, par exemple `C:\Users\<name>\Documents`
`%CommonSchemasFolder%`	`C:\ProgramData\Altova\Common2024\Schemas`
`%ApplicationWritableDataFolder%`	`C:\ProgramData\Altova`
`%AltovaCommonFolder%`	C:\Program Files\Altova\Common2024
`%DesktopFolder%`	Chemin complet vers le dossier Bureau de l'utilisateur actuel.
`%ProgramMenuFolder%`	Chemin complet vers le dossier Menu Programme pour l'utilisateur actuel.
`%StartMenuFolder%`	Chemin complet vers le dossier Démarrage pour l'utilisateur actuel.
`%StartUpFolder%`	Chemin complet vers le dossier Démarrage pour l'utilisateur actuel.
`%TemplateFolder%`	Chemin complet vers le dossier Modèle pour l'utilisateur actuel.
`%AdminToolsFolder%`	Chemin complet vers le répertoire de système de fichier qui stocke des outils administratifs pour l'utilisateur actuel.
`%AppDataFolder%`	Chemin complet vers le dossier Données d'Application pour l'utilisateur actuel.
`%CommonAppDataFolder%`	Chemin complet vers le répertoire de fichier contenant les données d'application pour tous les utilisateurs.
`%FavoritesFolder%`	Chemin complet du dossier Favoris pour l'utilisateur actuel.
`%PersonalFolder%`	Chemin complet vers le dossier Personnel pour l'utilisateur actuel.
`%SendToFolder%`	Chemin complet vers le dossier EnvoyerÀ pour l'utilisateur actuel.
`%FontsFolder%`	Chemin complet vers le dossier Polices Système.
`%ProgramFilesFolder%`	Chemin complet vers le dossier Fichiers Programme pour l'utilisateur actuel.
`%CommonFilesFolder%`	Chemin complet vers le dossier Fichiers Communs pour l'utilisateur actuel.
`%WindowsFolder%`	Chemin complet vers le dossier Windows pour l'utilisateur actuel.
`%SystemFolder%`	Chemin complet vers le dossier Système pour l'utilisateur actuel.
`%LocalAppDataFolder%`	Chemin complet vers le répertoire de fichier système qui sert en tant qu'archivage de données pour les applications locales (nonroaming).
`%MyPicturesFolder%`	Chemin complet vers le dossier MesPhotos.

11 Commandes de menu

Cette section décrit les commandes de menu de MapForce. Les commandes de menu suivantes sont disponibles :

11.1 Fichier

Cette rubrique recense toutes les commandes de menu disponibles dans le menu **Fichier**.

- Nouveau

  Crée un nouveau document de mappage. Dans les éditions Professional et Enterprise vous pouvez aussi créer un projet de mappage (`.mfp`).

- Ouvrir

  Ouvre le design de mappage précédemment ouvert (`.mfd`). Dans les éditions Professional et Enterprise, vous pouvez aussi ouvrir un projet de mappage (`.mfp`).

- Enregistrer/Enregistrer sous/Enregistrer tout

  L'option **Enregistrer** enregistre le mappage actuel actif sous son nom actuel. L'option **Enregistrer sous** vous permet d'enregistrer le mappage actuellement ouvert avec un différent nom. La commande **Enregistrer tout** enregistre tous les fichiers de mappage ouverts.

- Recharger

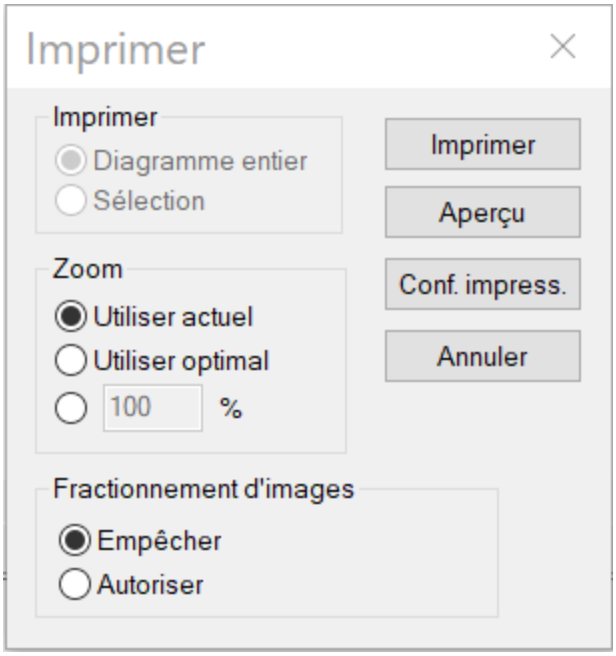
  Recharger le mappage actuel actif rétablit vos derniers changements.

- Fermer/Fermer tout

  La commande **Fermer** ferme le mappage actuel actif. La commande **Fermer tout** ferme tous les mappages actuels ouverts. Vous devez décider si vous souhaitez enregistrer les fichiers qui n'ont pas encore été enregistrés.

- Imprimer/Aperçu d'impression/Paramètres d'impression

  La commande **Imprimer** ouvre la boîte de dialogue **Imprimer** (*voir ci-dessous*) qui vous permet d'imprimer vos mappages. **Utiliser actuel** conserve le facteur de zoom actuellement défini pour le mappage. **Utiliser optimal** redimensionne le mappage pour qu'il s'adapte à la taille de la page. Vous pouvez aussi préciser le facteur de zoom numériquement. Les barres de déroulement du composant ne sont pas imprimées. Vous pouvez aussi préciser si vous voulez répartir les graphiques sur plusieurs pages ou non.



La commande **Aperçu** ouvre la même boîte de dialogue **Imprimer** avec les mêmes paramètres, tel que décrit ci-dessus. La commande **Paramètres d'impression** ouvre la boîte de dialogue **Paramètres d'impression** dans laquelle vous pouvez sélectionner une imprimante et configurer les paramètres papier.

- Valider mappage

  La commande **Valider mappage** vérifie si tous les mappages sont valides et affiche les messages d'information pertinents, les avertissements et les erreurs. Pour plus de détails, voir Validation[63].

- Paramètres de mappage

  Ouvre la boîte de dialogue des paramètres de mappage[74] où vous pouvez définir les paramètres spécifiques au document.

- Ouvrir gestionnaire d'identifiant (*Enterprise Edition*)

  Ouvre le **Gestionnaire d'identifiant** qui vous permet de gérer les identifiants requis dans les mappages qui réalisent une authentification HTTP de base ou une autorisation OAuth 2.0.

- Générer du code dans le langage sélectionné/Générer du code dans

  La commande **Générer Code dans langage sélectionné** génère du code dans le langage sélectionné dans la barre d'outils. La commande **Générer code dans <language>** vous permet de générer du code dans XSLT 1-3 (*toutes éditions*), XQuery, Java, C#, et C++ (*éditions Professional and Enterprise*). Sélectionner une commande ouvre la boîte de dialogue **Browse for Folder** dans laquelle vous avez besoin de sélectionner l'emplacement des fichiers générés. Le/s nom/s des fichiers est/sont défini/s dans la boîte de dialogue desParamètres de mappage[74].

Pour plus d'information sur les langages de transformation disponible, voir Langages de transformation[17]. Pour plus d'information sur le code généré, voir le Générateur de Code[65].

- Compiler sur fichier d'exécution MapForce Server (*éditions Professional et Enterprise*)

  Génère un fichier qui peut être exécuté par MapForce Server pour exécuter la transformation de mappage.

- Déployer sur FlowForce Server (*éditions Professional et Enterprise*)

  Déployez le mappage actuellement actif sur FlowForce Server.

- Générer documentation (*éditions Professional et Enterprise*)

  Génère la documentation de vos projets de mappage de manière très détaillée dans plusieurs formats de sortie.

- Fichiers récents

  Affiche la liste des fichiers ouverts le plus récemment.

- Quitter

  Quitte l'application. Vous devrez décider si vous souhaitez enregistrer les fichiers qui ne le sont pas encore.

11.2 Édition

Cette rubrique recense toutes les commandes de menu disponibles dans le menu **Édition**. La plupart des commandes dans ce menu deviennent actives lorsque vous regardez le résultat d'un mappage dans le volet **Sortie** ou dans le code aperçu, par exemple dans le volet **XSLT**.

- Annuler

  MapForce dispose d'un nombre illimité d'étapes "Annuler" que vous pouvez utiliser pour remonter vos étapes de mappage. Vous pouvez aussi utiliser la commande de barre d'outils  pour annuler les actions.

- Rétablir

  La commande rétablir vous permet de rétablir des commandes annulées précédemment. Vous pouvez retourner en arrière et en avant dans l'historique de rétablissement en utilisant ces deux commandes.

  Vous pouvez aussi utiliser la commande de barre d'outils  pour rétablir les actions.

- Recherche

  Vous permet de rechercher du texte spécifique dans tous les volets **XQuery** (*éditions Professional et Enterprise*), **XSLT**, **XSLT2**, **XSLT3**, et **Sortie**. Vous pouvez également effectuer la recherche en utilisant la commande de la barre d'outils  .

- Trouver suivant

  Cherche l'occurrence suivante du même string de recherche. Vous pouvez aussi rechercher la prochaine occurrence en utilisant  le bouton de la barre d'outils.

- Trouver précédent

  Cherche l'occurrence précédente du même string de recherche. Rechercher la prochaine occurrence est également possible avec la commande de barre d'outils  .

- Couper/ Copier/ Coller/ Supprimer

  Les commandes Édition des fenêtres standard, qui vous permettent de couper, coller, et supprimer tout composant ou fonctions visibles dans la fenêtre de mappage.

- Tout sélectionner

  Sélectionne tous les composants dans le volet **Mappage** ou le texte/code dans les volets **XQuery** (*éditions Professional et Enterprise*), **XSLT**, **XSLT2**, **XSLT3**, et **Sortie**.

11.3 Insérer

Cette rubrique recense toutes les commandes de menu disponibles dans le menu **Insérer**.

- Schéma XML/Fichier

  Ajoute un fichier de schéma XML ou le fichier d'instance au mappage. Si vous sélectionnez un fichier XML sans référence de schéma, MapForce peut générer un schéma XML correspondant 112. Si vous choisissez un fichier de schéma XML, vous serez invité à inclure en un fichier d'instance XML qui fournit les données pour l'aperçu. Vous pouvez également ajouter un fichier XML/XSD via la commande de la barre d'outils  .

- Base de données (*éditions Professional et Enterprise*)

  Ajoute un composant de base de données. Vous pouvez également ajouter un composant de base de données via la commande de la barre d'outils  . Dans l'édition MapForce Enterprise, vous pouvez aussi ajouter des bases de données NoSQL comme composants.

- EDI (*Enterprise Edition*)

  Ajoute un document EDI. Vous pouvez également ajouter un composant EDI via la commande de la barre d'outils  .

- Fichiers de texte (*éditions Professional et Enterprise*)

  Ajoute un document de fichier plat comme un fichier de texte CSV ou de longueur fixe. Vous pouvez également ajouter un fichier texte via la commande de la barre d'outils  . MapForce Enterprise Edition vous permet également de traiter les fichiers de texte avec FlexText.

- Fonction de Services web (*Enterprise Edition*)

  Ajouter un appel à un service Web générique. Vous pouvez également ajouter un service Web via le bouton de la barre d'outils  .

- Fichier Excel 2007+ (*Enterprise Edition*)

  Ajoute un fichier Microsoft Excel 2007+ (**`.xlsx`**). Si Excel 2007+ n#a pas été installé sur votre appareil, vous pouvez toujours mapper de ou vers des fichiers Excel 2007+. Dans ce cas, vous ne pouvez pas voir le résultat dans le volet **Sortie**, mais vous pouvez toujours enregistrer le résultat. Vous pouvez également ajouter un fichier Excel via la commande de la barre d'outils  .

- Document XBRL (*Enterprise Edition*)

  Ajoute une instance XBRL ou un document de taxonomie. Vous pouvez également ajouter un composant XBRL via la commande de la barre d'outils  .

- Schéma JSON/Fichier (*Enterprise Edition*)

Ajoute au schéma JSON ou un fichier. Vous pouvez également ajouter a composant JSON via la commande de la barre d'outils  .

- Fichier Protocol Buffers (*Enterprise Edition*)

  Ajoute un fichier binaire encodé dans un format Protocol Buffers. Vous pouvez également ajouter a fichiers binaires dans le format Protocol via la commande de la barre d'outils  .

- Insérer PDF document (*Enterprise Edition*)

  Ajoute un document PDF. Vous pouvez également insérer un document PDF via la barre d'outils.

- Insérer entrée

  Des composants Simple-Input peuvent être utilisés comme paramètres d'entrée qui sont pertinents à tout le mappage ou uniquement dans le contexte des fonctions définies par l'utilisateur. Pour plus d'informations, voir Simple Input 147 et Paramètres dans des UDF 209. Vous pouvez également Insérer un simple composant d'entrée utilisant la commande de la barre d'outils  .

- Insérer sortie

  Les composants de sortie simples peuvent être utilisés comme composants de sortie dans les mappages et comme paramètres des fonctions définies par l'utilisateur. Pour plus d'informations, voir Simple Output 154 et Paramètres dans des UDF 209. Vous pouvez également insérer un simple composant de sortie utilisant le commande de la barre d'outils  .

- Constante

  Insère une constante qui fournit les données fixes à un connecteur d'entrée. Vous pouvez sélectionner les types de données suivants : `string`, `nombre` et `tous les autres`. Vous pouvez également insérer une constante en utilisant la commande de la barre d'outils  .

- Variable

  Insère une variable 158, qui est l'équivalent d'une fonction définie par l'utilisateur régulière (non-inline). Une variable est un type de composant spécial utilisé pour stocker un résultat de mappage intermédiaire pour un traitement ultérieur. Vous pouvez également ajouter une variable en utilisant la commande de la barre d'outils  .

- Join (*éditions Professional et Enterprise*)

  Le composant Join vous permet de joindre des données en modes SQL et non-SQL. Vous pouvez également ajouter un composant Join utilisant la commande de la barre d'outils  .

- Trier: Nœuds/Lignes

  Insère un composant qui vous permet de trier des nœuds (voir Trier Nœuds/Lignes 170). Vous pouvez également ajouter un composant Tri utilisant la commande de la barre d'outils  .

- Filtrer: Nœuds/Lignes

  Insère un composant Filtre qui peut filtrer des données depuis toute autre structure de composant prise en charge par MapForce, y compris des bases de données. Pour plus d'informations, voir Filtres et Conditions 176. Vous pouvez également ajouter une filtre en utilisant la commande de la barre d'outils  .

- SQL/NoSQL-WHERE/ORDER (*éditions Professional et Enterprise*)

  Insère un composant qui vous permet de filtrer des données de base de données de manière conditionnelle. Vous pouvez également accéder un composant SQL/NoSQL-WHERE/ORDER via la commande de la barre d'outils  .

- Value-Map

  Insère un composant qui transforme une valeur d'entrée en une valeur de sortie en utilisant une table de consultation. Cela est utile lorsque vous devez mapper un ensemble de valeurs dans un autre ensemble de valeurs (par ex., des numéros de mois dans des noms de mois). Pour plus d'informations, voir Value-Maps 182. Vous pouvez également insérer une Value-Map en utilisant la commande de la barre d'outils  .

- Condition If-Else

  Insère une Condition If-Else qui se prête pour des scénarios dans lesquels vous devez traiter une valeur simple par condition. Pour plus d'informations, voir Filtres et Conditions 176. Vous pouvez également ajouter une condition If-Else utilisant la commande de la barre d'outils  .

- Exception (*éditions Professional et Enterprise*)

  Le composant d'exception vous permet d'interrompre une procédure de mappage lorsqu'une condition spécifique est remplie. Vous pouvez également ajouter un composant d'Exception utilisant la commande de la barre d'outils  . Dans MapForce Enterprise Edition, ce composant vous permet de définir les messages Fault dans les projets de mappage WSDL.

- Commenter

  Cette commande de menu vous permet d'insérer des commentaires style pense-bête comme composants amovibles. Pour les détails, voir Commentaires 34. Vous pouvez également ajouter des commentaires en cliquant la commande de barre d'outils  .

11.4 Composant

Cette rubrique recense toutes les commandes de menu disponibles dans le menu **Composant**.

- Modifier Élément racine

  Vous permet de changer l'élément racine du document d'instance XML.

- Éditer une définition de Schéma dans XMLSpy

  Afin de pouvoir éditer un schéma dans Altova XMLSpy, vous devez cliquer un composant XML, puis sélectionnez l'option **Éditer définition de Schéma dans XMLSpy**.

- Éditer Configuration FlexText (*Enterprise Edition*)

  Cette commande vous permet d'éditer un fichier FlexText.

- Ajouter/Supprimer/Éditer des objets de base de données (*éditions Professional et Enterprise*)

  Vous permet d'ajouter, de supprimer et de changer des objets de base de données dans le composant de base de données.

- Créer un mappage pour EDI X12 997 (*Enterprise Edition*)

  Le X12 997 Functional Acknowledgment rapporte le statut de l'interchange EDI. Toutes les erreurs rencontrées pendant le traitement du document y sont rapportées. MapForce peut générer automatiquement un document X12 997 que vous pourrez envoyer au destinataire.

- Créer un mappage pour EDI X12 999 (*Enterprise Edition*)

  Le X12 999 Implementation Acknowledgment Transaction Set rapporte la non-conformité du guide d'implémentation HIPAA ou les erreurs d'application. MapForce peut générer automatiquement le composant X12 999 dans le mappage et créer automatiquement les connexions de mappage nécessaires.

- Actualiser*éditions Professional et Enterprise*)

  Recharge la structure du composant de base de données actuellement active.

- Ajouter double entrée Avant/Après

  Insère une copie de l'item sélectionné avant/aprés l'item sélectionné. L'entrée dupliquée ne peut pas être utilisée comme source de données. Pour plus d'information, voir Dupliquer l'entrée 40.

- Supprimer doublon

  Supprime un item dupliqué.

- Instructions de traitement/Commentaire

  Cette option vous permet d'insérer des commentaires et instructions de traitement 122 dans les composants XML.

- Écrire du contenu en tant que section CDATA

Cette commande crée une section CDATA[122] qui est utilisée pour représenter des parties d'un document comme données de caractère qui devraient normalement être interprétés comme balise.

- Actions de Table de base de données (*éditions Professional et Enterprise*)

  Vous permet de configurer l'insertion de base de données, mettre à jour et supprimer des actions et d'autres options d'enregistrements de base de données.

- Requête de base de données (*éditions Professional et Enterprise*)

  Crée une instruction SELECT sur la base de la table/champ sur lequel vous avez cliqué dans le composant de la base de données. Cliquez sur une table/un champ pour rendre active cette commande, et l'instruction SELECT est placée automatiquement dans la **fenêtre** Select.

- Aligner arborescence à gauche

  Rend l'arborescence d'un composant left-justified.

- Aligner arborescence à droite

  Rend l'arborescence d'un composant right-justified.

- Éditer les Commentaires

  Si vous avez un composant commentaire dans votre mappage, vous ppouvez l'éditer, en cliquant dessus et en sélectionnant la commande **Éditer Commentaire**. En alternative, vous pouvez double-cliquer à l'intérieur du composant commentaire et éditer le texte directement dans la case des commentaires. Pour plus d'information sur les Composants commentaires et leurs types, voir Commentaire[34].

- Propriétés

  Affiche les paramètres d'un composant actuellement sélectionné. Voir Changer les paramètres de composant[39].

11.5 Connexion

Cette rubrique recense toutes les commandes de menu disponibles dans le menu **Composant**.

- Auto-connexion des enfants correspondants

  Activer/désactiver l'option **Auto-connexion des enfants correspondants**. Pour plus d'information sur les connexions et leurs types, voir Connexions 46.

- Paramètres pour connecter des enfants correspondants

  Vous aide à définir des connexions d'enfants correspondants. Pour les détails, voir Connexions d'enfants correspondants 52.

- Connecter les enfants correspondants

  Cette commande vous permet de créer de multiples connexions pour des items avec les mêmes noms dans les composants source et cible. Les paramètres que vous définissez dans cette boîte de dialogue s'appliquent si la commande de la barre d'outils  (**Auto-connexion des éléments enfants**) a été activée. Pour les plus d'informations, voir Connexions d'enfants correspondants 52.

- Orienté vers la cible (Standard)

  Change le type de connecteur sur un mappage standard. Pour plus d'information, voir Connexions Target-driven vs. source-driven 50.

- Copier-tout (Copier les items enfants)

  Crée des connexions popur tous les items enfant correspondants. Le principal avantage des connexions copier-tout est qu'elles simplifient visuellement l'espace de travail du mappage : Une connexion, représentée par une ligne épaisse, est créée à la place de connexions multiples. Pour plus de détails, voir Connexions copier tout 55.

- Orienté vers la source (contenu mixte)

  Change un type de connexion à une connexion orientée vers la source qui vous permet de mapper automatiquement le contenu mixte (nœuds texte et enfant) dans le même ordre que dans le fichier XML *source*. Pour plus d'informations, voir Connexions orientées vers la source 50.

- Propriétés

  Ouvre la boîte de dialogue **Paramètres de connexion** qui vous permet de définir des types de connexion et des paramètres d'annotation. Pour plus d'informations, voir Paramètres de connexion 56.

11.6 Fonction

Cette rubrique recense toutes les commandes de menu disponibles dans le menu **Fonction**.

- Créer une fonction définie par l'utilisateur

  Créer une (UDF) fonction définie par l'utilisateur[203]. Vous pouvez également créer l'UDF en utilisant la commande de la barre d'outils  .

- Créer une fonction définie par l'utilisateur depuis la sélection

  Crée une fonction définie par l'utilisateur basée sur les éléments sélectionnés actuellement dans la fenêtre de mappage. Pour les détails, voir Créer des UDF[206]. Vous pouvez également créer un UDF de la sélection utilisant la commande de la barre d'outils  .

- Paramètres de fonction

  Ouvre la boîte de dialogue **Éditer la fonctions définie par l'utilisateur** qui vous permet de changer des paramètres de l'UDF. Pour les détails, voir Éditer des UDF[204].

- Supprimer la fonction

  Supprime la fonction définie par l'utilisateur actuellement active si vous travaillez dans un contexte qui permet ceci.

- Insérer entrée

  Des composants Simple-Input peuvent être utilisés comme paramètres d'entrée qui sont pertinents à tout le mappage ou uniquement dans le contexte des fonctions définies par l'utilisateur. Pour plus d'informations, voir Simple Input[147] et Paramètres dans des UDF[209]. Vous pouvez également Insérer un simple composant d'entrée utilisant la commande de la barre d'outils  .

- Insérer sortie

  Les composants de sortie simples peuvent être utilisés comme composants de sortie dans les mappages et comme paramètres des fonctions définies par l'utilisateur. Pour plus d'informations, voir Simple Output[154] et Paramètres dans des UDF[209]. Vous pouvez également Insérer un simple composant de sortie utilisant le bouton de la barre d'outils  .

11.7 Sortie

Cette rubrique recense toutes les commandes de menu disponibles dans le menu **Sortie**.

- XSLT 1.0/XSLT 2.0/XSLT 3.0/XQuery/Java/C#/C++/Built-In

  Définit le langage de transformation dans lequel le mappage doit être exécuté. La sélection du langage de transformation dépend de votre édition de MapForce. Pour des détails, voir Langages de transformation[17] . Vous pouvez aussi sélectionner des langages de a transformation dans la barre d'outils.

- Valider le fichier Sortie

  Valide le fichier XML de sortie par rapport à un schéma référencé. Voir Validation[63].

- Enregistrer le fichier de sortie

  Enregistre les données dans le volet **Sortie** dans un fichier.

- Enregistrer tous les fichiers de sortie

  Enregistre tous les fichiers de sortie générés des mappages dynamiques[400]. Voir le Tutoriel 4[102].

- Régénérer Sortie

  Régénère les données visibles dans le volet **Sortie**.

- Exécuter SQL/NoSQL-Script (*éditions Professional et Enterprise*)

  Si un script SQL/NoSQL est actuellement visible dans le volet **Sortie**, le script exécute le mappage vers une base de données cible, prenant les actions de table définies en compte.

- Insérer/Supprimer signet

  Insère/supprime un signet à la position du curseur dans le volet **Sortie**.

- Signet suivant/précédent

  Navigue vers le signet suivant/précédent dans le volet **Sortie**.

- Supprimer tous les signets

  Supprime tous les signets définis actuellement dans le volet **Sortie**.

- Texte XML Pretty-Print

  Reformate votre document XML dans le volet **Sortie** pour que le document ait un affichage structuré : Chaque nœud enfant est décalé de son parent par un seul caractère de tabulation. Dans le volet **Sortie**, les paramètres de taille de l'onglet définis dans le dialogue Paramètres du Mode Texte[67] (groupe d'onglets) prennent effet.

- Paramètres Affichage Texte

  Affiche la boîte de dialogue **Paramètres du Mode Texte** qui vous permet de personnaliser les paramètres du Mode Texte dans le volet **XQuery** (*éditions Professional et Enterprise*), le volet **Sortie** et le

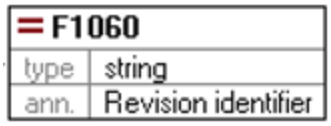
volet **XSLT**. Le dialogue affiche également les raccorcis clavieractuellement définis. Pour plus d'informations, voir Fonctions de Mode Texte[67].

11.8 Affichage

Cette rubrique recense toutes les commandes de menu disponibles dans le menu **Affichage**.

- Afficher Annotations

  Afficher les annotations dans le composant. Vous pouvez aussi activer cette option en cliquant sur la touche de la barre d'outils  . Si l'icône **Afficher Types** est aussi active, les deux ensembles d'information sont affichés sous forme de grille (*voir la capture d'écran ci-dessous*). Vous pouvez aussi utiliser des annotations pour libeller les connexions. Pour les détails, voir Paramètres de connexion 58.

- Afficher Types

  Afficher les types de données dans le composant. Vous pouvez aussi activer cette option en cliquant sur la touche de la barre d'outils  . Si l'icône **Afficher Annotations** est aussi active, alors deux ensembles d'information sont enregistrements affiché sous forme de grille (*voir Afficher Annotations*).

- Afficher la Bibliothèque dans l'en-tête Fonction

  Affiche le nom de bibliothèque dans l'en-tête de la fonction. Vous pouvez aussi activer cette option en cliquant sur la touche de la barre d'outils  .

- Afficher astuces

  Lorsque vous placez le curseur au-dessus d'un en-tête de fonction, vous verrez une info-bulle résumant ce que cette fonction fait. Avec l'option **Afficher astuces** activée, vous pouvez aussi voir l'information sur les types de données dans un composant.

- Options d'affichage XBRL(*Enterprise Edition*)

  MapForce vous permet de configurer les paramètres XBRL suivants :

  - Le langage de libellé des items XBRL et leurs annotations
  - Les rôles de libellé préférés pour les noms d'item XBRL
  - Le type spécifique de rôles de libellé des annotations pour les items XBRL
  - Packages de Taxonomie XBRL personnalisés

- Afficher les connecteurs de composant sélectionnés/ connexions depuis la source vers la cible

  Ces options vous permettent de surligner les connexions de manière sélective. Pour savoir comment ces options fonctionnent, voir Connexions 48.

- Zoom

Ouvre le dialogue **Zoom**. Vous pouvez saisir le facteur de manière numérique et faire glisser le curseur pour changer le facteur zoom de manière interactive.

- Précédent/ Suivant

  Les commandes **Retour** et **Suivant** vous permettent de basculer entre les mappages précédents ou suivants sur lesquels vous avez travaillé, relatifs au mappage actuellement ouvert.

- Barre de statut

  Active/désactive la **barre de statut** visible en-dessous de la fenêtre **Messages**.

- Bibliothèques/Gérer les Bibliothèques

  Cliquez sur **Bibliothèques** pour activer/désactiver la fenêtre **Bibliothèques**. Cliquez sur **Gérer les bibliothèques** pour activer/désactiver la fenêtre **Gérer les bibliothèques**.

- Messages

  Active/désactive la fenêtre des Messages [25]. Lorsque le code est généré, la fenêtre **Messages** est automatiquement activée pour montrer le résultat de validation.

- Aperçu

  Active/désactive la fenêtre d'aperçu [24]. Faites glisser le rectangle pour naviguer à travers le mappage.

- Fenêtre de projet (*éditions Professional et Enterprise*)

  Active/désactive la fenêtre **Projet**.

- Fenêtres Débogage (*éditions Professional et Enterprise*)

  Le mode de débogage vous permet d'analyser le contexte responsable dans lequel une valeur particulière est produite. Cette information est disponible directement sur le mappage et dans les fenêtres **Valeurs**, **Contexte** et **Points d'arrêt**.

11.9 Outils

Cette rubrique recense toutes les commandes de menu disponibles dans le menu **Outils**.

- Ressources globales

  Ouvre la boîte de dialogue **Gérer les Ressources globales** où vous pouvez ajouter, éditer et supprimer des paramètres applicables à travers les multiples applications d'Altova. Pour plus d'informations, voir Ressources globales Altova 430.

- Configuration active

  Vous permet de sélectionner une configuration de ressource active actuelle depuis une liste de configurations. Pour créer et configurer différents types de ressources globales, voir Ressources globales Altova 430.

- Créer mappage inversé

  Crée un mappage contrepassé du mappage actuellement actif, ce qui signifie que le composant source devient un composant cible et le composant cible devient la source. Notez que seuls les connexions directes entre des composants sont retenues dans le mappage contrepassé. Il est probable que le nouveau mappage ne sera pas valide ou approprié pour un aperçu dans le volet **Sortie**. Pour cette raison, le nouveau mappage devrait requérir l'édition manuelle.

  Les données suivantes sont retenues :

  - Connexions directes entre des composants
  - Connexions directes entre des composants dans un mappage enchaîné
  - Le type of connexion 49 : Standard, Contenu mixte, Copier-tout
  - Paramètres de composant de passage
  - Composants de base de données (*éditions Professional et Enterprise*)

  Les données suivantes ne sont *pas* retenues :

  - Connexions via les fonctions, filtres, etc.
  - Fonctions définies par l'utilisateur
  - Composants de service Web (*Enterprise Edition*)

- Gestionnaire de taxonomie XBRL (*Enterprise Edition*)

  Le gestionnaire de taxonomie XBRL est un outil qui permet d'installer et de gérer des taxonomies XBRL.

- Gestionnaire de schéma XML

  Le Gestionnaire de schéma XML est un outil qui propose un moyen centralisé d'installer et de gérer des schémas XML (DTD pour XML et Schémas XML) pour une utilisation sur toutes les applications activées par XBRL d'Altova. Pour plus d'informations, voir le Gestionnaire de schéma 129.

- Personnaliser

  Cette option vous permet de personnaliser l'interface d'utilisateur graphique de MapForce. Ceci inclut de montrer/ cacher les barres d'outils de même que de personnaliser les menus 465 et les raccourcis de clavier 466.

- Restaurer les Barres d'outils et Fenêtres

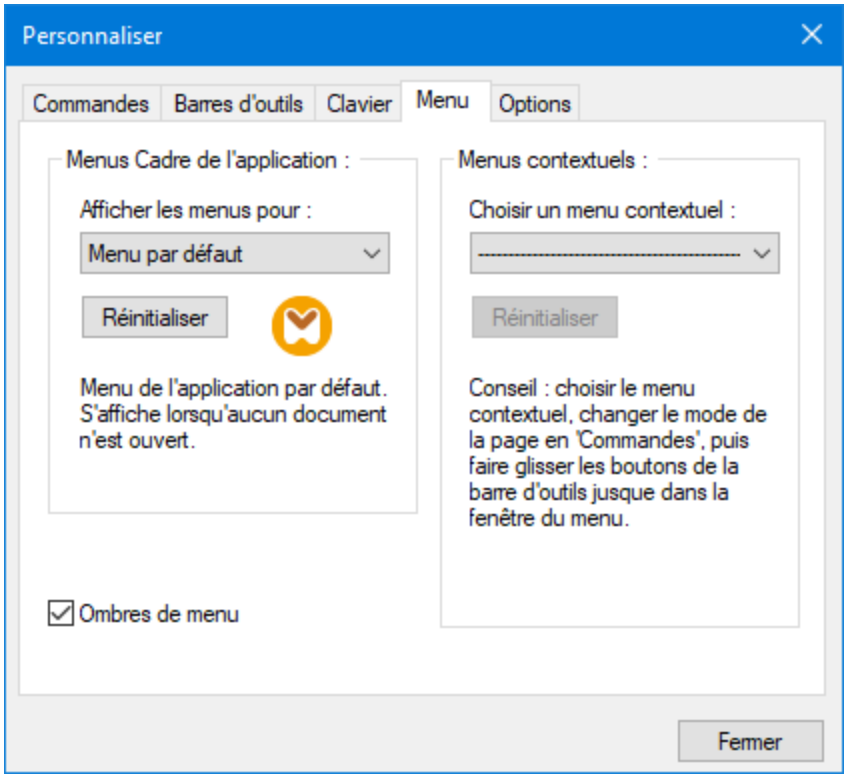
  Restaure les barres d'outils, les fenêtres d'assistants à la saisie, les fenêtres ancrées, etc. à leur affichage par défaut. Vous devez redémarrer MapForce pour que les changements prennent effet.

- Options

  Ouvre la boîte de dialogue **Options** qui vous permet de changer les paramètres par défaut de MapForce. Pour plus d'informations, voir Options 469.

11.9.1 Personnaliser les menus

Vous pouvez personnaliser des menus standard de MapForce de même que des menus contextuels (par ex., pour ajouter, modifier ou supprimer les commandes). Vous pouvez rétablir vos changements à l'état par défaut (**Rétablir**). Pour personnaliser les menus, allez à **Outils | Personnaliser** et cliquez sur l'onglet **Menu** (*voir la capture d'écran ci-dessous*).



Menu par défaut vs. MapForce Design
La barre *Menu par Défaut* est affichée lorsqu'aucun document n'est ouvert dans la fenêtre principale. La barre de menu *MapForce Design* est la barre de menu qui est affichée lorsqu'un ou plusieurs mappage sont ouverts. Chaque barre de menu peut être personnalisée séparément. Les changements de personnalisation effectués à une barre de menu n'affectent pas l'autre.

Pour personnaliser une barre de menu, la choisir depuis la liste déroulante *Afficher Menus pour*. Ensuite, cliquez sur l'onglet **Commandes** et glissez les commandes depuis le champ de liste *Commandes* dans la barre de menu ou dans un des menus.

Supprimer les commandes depuis les menus
Pour supprimer un menu ou une commande entière depuis un menu, procéder comme suit :

1. Sélectionnez *Default Menu* ou *MapForce Design* depuis la liste déroulante *Afficher menus pour*.
2. Lorsque le dialogue **Personnaliser** est ouvert, choisissez une commande de la barre d'outils que vous souhaitez supprimer ou une commande que vous souhaitez supprimer depuis un des menus.
3. Glissez la commande de la barre d'outils ou la commande depuis le menu. En alternative, cliquez avec la touche de droite sur la commande de la barre d'outils ou la commande de menu et sélectionnez **Supprimer**.

Vous pouvez réinitialiser toute barre de menu à son état d'installation originale en le sélectionnant depuis la liste déroulante *Afficher menus pour* et cliquez sur le bouton **Réinitialiser**.

Personnaliser les menus contextuels
Les menus contextuels qui apparaissent lorsque vous cliquez avec la touche de droite sur certains objets dans l'interface de l'application. Chacun de ces menus contextuels peuvent être personnalisés en procédant comme suit :

1. Choisissez le menu contextuel depuis la liste déroulante *Sélectionner un menu contextuel*. Ceci ouvre un menu contextuel respectif.
2. Ouvrez l'onglet **Commandes** et glissez une commande depuis la zone de liste *Commandes* dans le menu contextuel.
3. Pour supprimer une commande depuis le menu contextuel, cliquez avec la touche de droite sur cette commande et sélectionnez **Supprimer**. En alternative, glissez la commande hors du menu contextuel.

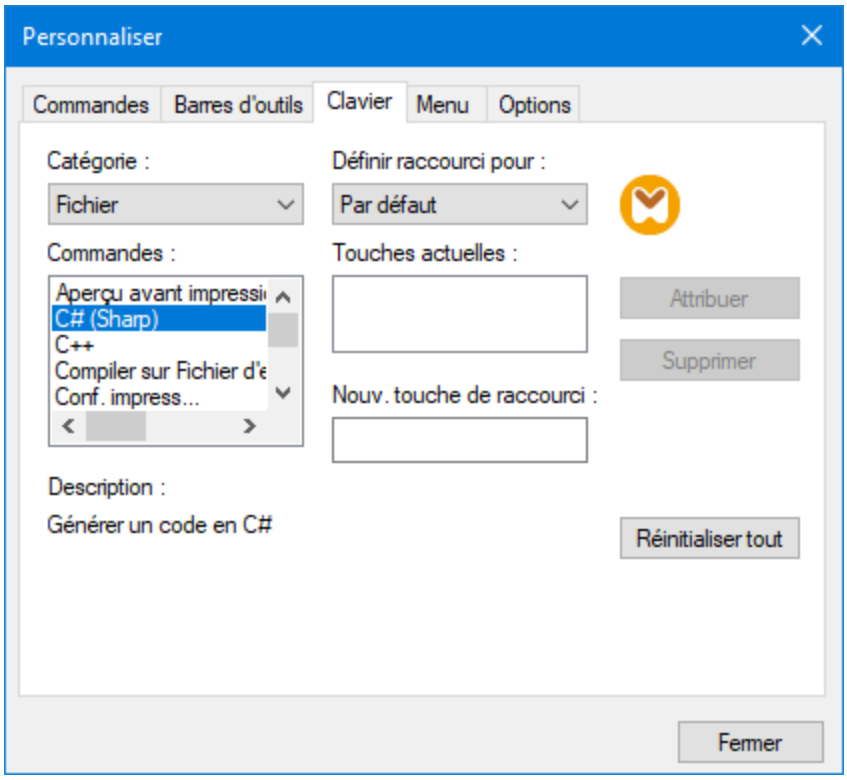
Vous pouvez réinitialiser un menu contextuel dans son état par défaut en le choisissant dans la liste déroulante *Sélectionner le menu contextuel* cliquer sur la touche **Réinitialiser**.

Ombres de menu
Sélectionnez la case à cocher *Ombres de menu* pour ajouter des ombres à tous les menus.

11.9.2 Personnaliser les raccourcis de clavier

Vous pouvez définir ou changer les raccourcis de clavier dans MapForce comme suit : Sélectionnez **Outils | Personnaliser** et cliquez sur l'onglet **Clavier**. Pour attribuer un nouveau raccourci à une commande, suivez les étapes suivantes :

1. Sélectionnez la commande **Outils | Personnaliser** et cliquez sur l'onglet **Clavier** (*voir la capture d'écran ci-dessous*).
2. Cliquez sur la zone de liste modifiable *Catégorie* pour sélectionner le nom du menu.
3. Dans la zone de liste *Commandes*, sélectionnez la commande à laquelle vous souhaitez attribuer dans un nouveau raccourci.
4. Saisissez de nouvelles clés de raccourcis dans la zone de texte *Appuyer sur Nouvelle clé de raccourci* et cliquez sur **Attribuer**.



Pour effacer la saisie dans la zone de texte *Appuyer sur Nouvelle clé de raccourci*, appuyez sur les clés de commandes : **Ctrl**, **Alt** ou **Shift**. Pour supprimer un raccourci, cliquez sur le raccourci que vous souhaitez supprimer dans la zone de liste *Clés actuelles* et cliquez sur **Supprimer**.

Note : Le *Set accelerator for* n'a pas de fonction pour le moment.

Raccourcis clavier

Par défaut, MapForce attribue les raccourcis de clavier suivants:

F1	Menu d'aide
F2	Signet suivant (dans la fenêtre de sortie)
F3	Trouver suivant
F10	Activer la barre de menu
Num +	Agrandir le nœud d'item actuel
Num -	Réduire le nœud d'item
Num *	Agrandir tout depuis le nœud d'item actuel
CTRL + TAB	Passe entre les mappages ouverts
CTRL + F6	Feuilleter dans les fenêtres ouvertes
CTRL + F4	Ferme le document de mappage actif
Alt + F4	Ferme MapForce
Alt + F, F, 1	Ouvre le dernier fichier
Alt + F, T, 1	Ouvre le dernier projet

Ctrl + N	Nouveau fichier
Ctrl + O	Ouvrir fichier
Ctrl + S	Enregistrer fichier
Ctrl + P	Imprimer fichier
CTRL + A	Sélectionner tout
Ctrl + X	Couper
CTRL + C	Copier
Ctrl + V	Coller
Ctrl + Z	Annuler
CTRL + Y	Rétablir
Suppr	Supprimer composant (avec invitation)
Shift + Del	Supprimer composant (sans invitation)
CTRL + F	Recherche
F3	Trouver suivant
Shift + F3	Trouver précédent

Touches fléchées

(haut / bas)	Choisir item de composant suivant
Échap	Abandonner éditions/fermer dialogue
Return	Confirme une sélection

Raccourcis fenêtre sortie

CTRL + F2	Insérer Supprimer/Signet
F2	Signet suivant
SHIFT + F2	Signet précédent
CTRL + SHIFT + F2	Supprimer tous les signets

Raccourcis Zoom

CTRL + roue de la souris en avant	Zoom avant
CTRL + roue de la souris en arrière	Zoom arrière
CTRL + 0 (Zéro)	Réinitialiser Zoom

11.10 Options

Vous pouvez modifier préférences générales et autres dans MapForce sortie en sélectionnant la commande **Outils | Options**. Les options disponibles sont les suivantes :

- Généralités

Dans la section *Généralités*, vous pouvez définir les options suivantes :

- *Afficher logo | Afficher sur démarrage :* Affiche ou masque une image (splash screen) lorsque MapForce démarre.

- La section *Aperçu Mappage* vous permet de définir les paramètres suivants:

  - Vous pouvez activer/désactiver l'arrière-plan dégradé dans le volet Mappage (*Afficher l'arrière-plan dégradé*).
  - Vous pouvez limiter l'affichage des annotations aux lignes N (*Limiter l'affichage d'annotation*). Par exemple, si vous avez défini cette option à 2, et que votre texte d'annotation contient 3 lignes, uniquement les deux premières lignes du texte d'annotation seront affichées dans le mappage. Ce paramètre s'applique également aux instructions SELECT visibles dans un composant.
  - Vous pouvez aussi limiter l'affichage du (commentaire du composant 32) aux lignes N (*Limiter l'affichage du commentaire*) Par exemple, si vous avez limité l'affichage du commentaire à 1 ligne et que votre commentaire contient plus d'une ligne, la boîte de commentaire affichera uniquement la première ligne. Définir la propriété à `0` masquera entièrement l'affichage des commentaires. Notez que l'option *Limiter l'affichage du commentaire* n'a pas d'effet sur les composants de commentaire 34.

- *Encodage par défaut pour les nouveaux composants.*

  Nom d'encodage : L'encodage par défaut pour les nouveaux fichiers XML peut être défini en sélectionnant une option depuis la liste déroulante. Si un encodage à deux-ou-quatre-byte est sélectionné comme encodage par défaut (par ex., UTF-16, UCS-2 ou UCS-4), vous pouvez aussi choisir entre un tri little-endian et big-endian byte. Ce paramètre peut également être modifié individuellement pour chaque composant(voir Changer paramètres du composants 39).

  Tri d'octets : Quand un document à encodage de caractère two-byte ou four-byte est enregistré, le document peut être enregistré soit avec un tri d'octets little-endian ou big-endian. Vous pouvez également spécifier si une marque de tri d'octets devrait être incluse ou non.

- *Paramètres d'aperçu :* L'option *Utiliser la marche d'arrêt d'exécution* définit une marche d'arrêt d'exécution lorsque vous afficher le résultat de mappage dans le volet **Output**.

- *Sur activation du Volet de sortie :* Vous pouvez générer la sortie vers les fichiers temporaires ou écrire la sortie directement dans un fichier de sortie (*voir ci-dessous*).

  Générer la sortie vers les fichiers temporaires : Il s'agit de l'option par défaut. Si le chemin de fichier de sortie contient des dossiers qui n'existent pas encore, MapForce créera ces dossiers. Pour les éditions MapForce Professional et Enterprise : Si vous avez l'intention de déployer le mappage vers un serveur pour l'exécution, tout répertoire dans le champ doit exister sur le serveur ; sinon, une erreur d'exécution se produira.

Écrire directement sur les fichiers de sortie finaux : Si le chemin de fichier de sortie contient des dossiers qui n'existent pas encore, une erreur de mappage se produira. Cette option écrase tout fichier de sortie existant sans demander de confirmation supplémentaire.

- *Afficher le texte par étapes de N millions de caractères :* Spécifie la taille maximum du texte affiché dans le volet de **Sortie** lorsque vous consultez des mappages qui génèrent des grands fichiers XML et texte. Si le texte de sortie dépasse cette valeur, vous devrez cliquer sur une touche **Charger plus** pour charger le bloc suivant. Pour plus d'informations, voir Consulter et valider le sortie 63.

Éditer

Dans la section *Édition*, vous pouvez définir les options suivantes :

- *Aligner les composants sur glissage de souris :* Spécifiez si des composants ou des fonctions doivent être alignés avec d'autres composants, alors que vous les glissez dans la fenêtre Mappage. Pour plus d'informations, voir Aligner des composants 40.

- *Suppression de composant intelligent :* MapForce vous permet de garder les connexions même après avoir supprimé quelques composants de transformation 32. Garder les connexions peut être particulièrement utile avec de multiples connexions enfant car vous ne devrez pas restaurer chaque connexion simple enfant manuellement après avoir supprimé le composant de transformation. Pour les détails, voir Garder des connexions après avoir supprimé des composants 61.

Messages

La section *Messages* vous permet d'activer des notifications de message telles que la suggestion de connecter des items ancêtre, informant sur la création de multiples composants, etc.

Génération (*éditions Professional et Enterprise*)

La section *Génération* vous permet de définir des paramètres pour la génération du program-code et les fichiers d'exécution de MapForce Server.

Java

Vous devrez éventuellement ajouter un chemin Java VM personnalisé, par exemple si vous utilisez une machine virtuelle Java qui ne possède pas de programme d'installation et ne crée pas d'entrées de registre (par exemple, OpenJDK d'Oracle). Vous pourrez également définir ce chemin si vous souhaitez contourner tout chemin Java VM détecté automatiquement par MapForce. Pour plus de détails, voir Java 472.

XBRL (*Enterprise Edition*)

MapForce vous permet de configurer les paramètres (dans toute l'application) XBRL :

- Le langage de libellé des items XBRL et leurs annotations
- Les rôles de libellé préférés pour les noms d'item XBRL
- Le type spécifique de rôles de libellé des annotations pour les items XBRL
- Packs de taxonomie XBRL

- Débogueur (*éditions Professional et Enterprise*)

  Dans la section *Débogueur*, vous pouvez définir les paramètres de débogage suivants :

  - *Longueur maximum des valeurs stockées :* Définit la longueur du string des valeurs affichées dans la fenêtre **Valeurs** (au moins 15 caractères). Veuillez noter que le réglage de la longueur de stockage à une valeur élevée peut vider la mémoire système disponible.

  - *Garder l'historique du traçage complet :* Instruit MapForce de garder l'historique de toutes les valeurs traitées par toutes les connecteurs de tous les composants dans le mappage pour la durée du débogage. Si cette option est activée, toutes les valeurs traitées par MapForce depuis le début de l'exécution de débogage seront stockées dans la mémoire et seront disponibles pour une analyse dans la fenêtre **Valeurs**, jusqu'à ce que vous cessiez de le déboguer. Il n'est pas recommandé d'activer cette option si vous êtes en train de déboguer des mappages nécessitant beaucoup de données, étant donné qu'elles pourraient ralentir l'exécution de débogage et vider la mémoire système disponible. Si cette option est désactivée, MapForce ne gardera que l'historique de trace la plus récente pour les nœuds liés à la position d'exécution actuelle.

- Base de données (*éditions Professional et Enterprise*)

  Dans la section *Base de données*, vous pouvez définir les paramètres de requête de la base de données.

- Proxy de réseau

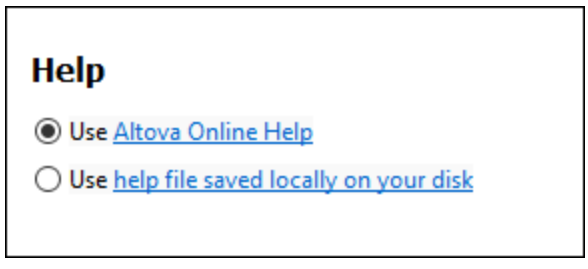
  La section *Proxy de réseau* vous permet de configurer des paramètres de proxy personnalisés. Ces paramètres affectent la manière dont l'application se connecte à Internet. Par défaut, l'application utilise les paramètres proxy du système, dans la plupart des cas, vous n'aurez donc pas à changer les paramètres de proxy. Pour plus de détails, voir Proxy de réseau 474.

- Aide

  MapForce fournit de l'Aide (le manuel utilisateur) en deux formats :

  - Aide en ligne, sous format HTML, qui est disponible sur le site web d'Altova. Afin d'accéder à l'Aide en ligne, vous aurez besoin d'un accès Internet.
  - Un fichier Aide sous format PDF, est installé sur votre machine quand vous installez MapForce. Il est appelé `MapForce.pdf` et est situé dans le dossier d'application (dans le dossier de fichiers de programme). Si vous n'avez pas accès à Internet, vous pouvez toujours ouvrir ce fichier Aide enregistré localement.

  L'option Aide (*capture d'écran ci-dessous*) vous permet de sélectionner lequel des deux formats s'ouvre quand vous cliquez sur la commande **Aide (F1)** dans le menu **Aide**.

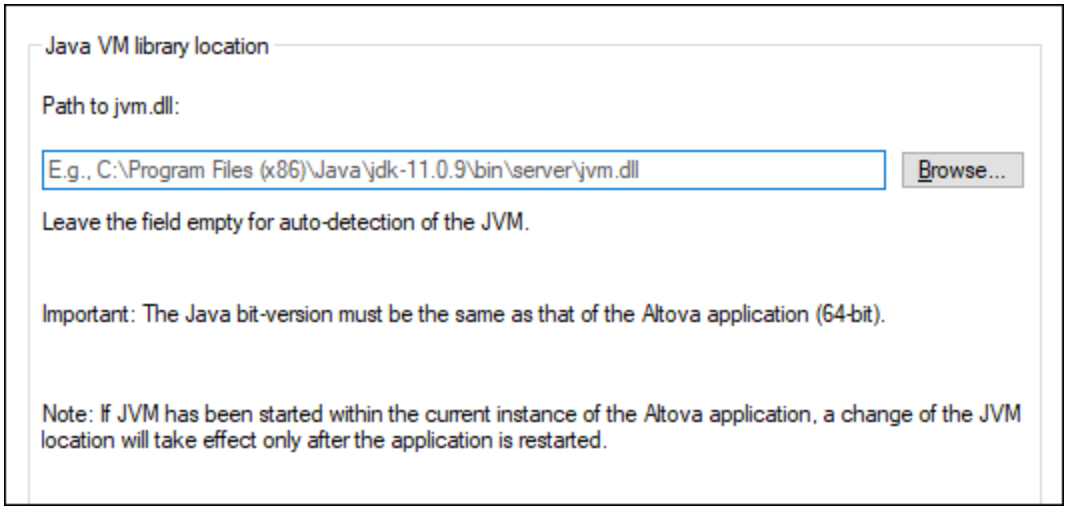
  Vous pouvez modifier cette option à tout moment pour la nouvelle sélection. Les liens dans cette section

(*voir capture d'écran ci-dessus*) ouvre le format Aide respectif.

11.10.1 Java

Dans l'onglet *Java* (*voir la capture d'écran ci-dessous*), vous pouvez saisir en option un chemin vers une Java VM (Machine Virtuelle) sur votre système de fichier. Veuillez noter que le fait d'ajouter un chemin Java VM personnalisé n'est pas toujours nécessaire. Par défaut, MapForce tente de détecter le chemin Java VM automatiquement en lisant (dans cet ordre) le registre Windows et la variable d'environnement JAVA_HOME. Le chemin personnalisé ajouté dans ce dialogue prendra la priorité sur tout autre chemin Java VM détecté automatiquement.

Vous devrez éventuellement ajouter un chemin Java VM personnalisé, par exemple si vous utilisez une machine virtuelle Java qui ne possède pas de programme d'installation et ne crée pas d'entrées de registre (par exemple, OpenJDK d'Oracle). Vous pourrez également définir ce chemin si vous souhaitez contourner, pour une raison quelconque, tout chemin Java VM détecté automatiquement par MapForce.

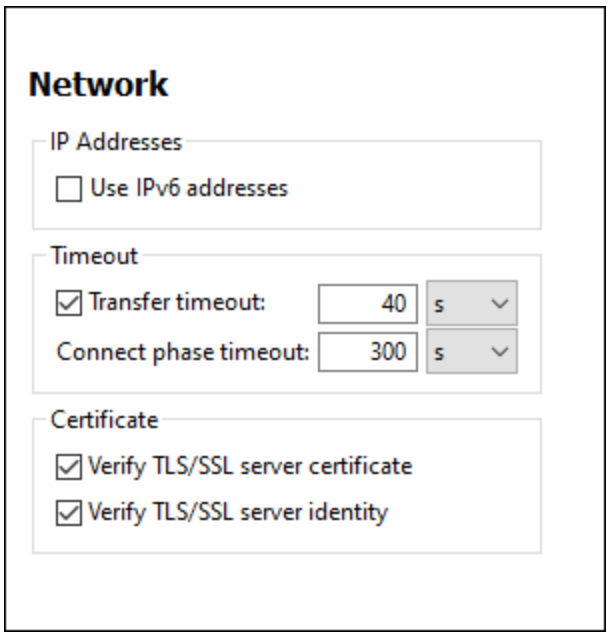


Veuillez noter les points suivants :

- Le chemin Java VM est partagé entre les applications de desktop (pas serveur) Altova. Par conséquent, si vous le modifiez dans une application, il s'appliquera automatiquement à toutes les autres applications Altova.
- Le chemin doit pointer vers le fichier **`jvm.dll`** provenant du répertoire **`\bin\server`** ou **`\bin\client`**, par rapport au répertoire sur lequel le JDK a été installé.
- La plateforme MapForce (32-bit, 64-bit) doit être la même que celle du JDK.
- Une fois avoir modifié le chemin Java VM, vous devrez éventuellement redémarrer MapForce pour que les nouveaux paramètres prennent effet.

11.10.2 Réseau

La section **Réseau** (*voir la capture d'écran ci-dessous*) vous permet de configurer les paramètres de proxy de réseau.



Adresses IP
Lorsque les noms d'hôte résolvent plus d'une adresse dans les réseaux mixtes IPv4/IPv6, sélectionner cette option fait que les adresses IPv6 sont utilisées. Si l'option n'est pas sélectionnée dans de tels environnements, alors les adresses IPv4 sont utilisées.

Délai d'expiration
- *Délai d'expiration du transfert :* Si cette limite est atteinte pour deux packs de données consécutifs d'un transfert (envoyés ou reçus), alors le transfert entier est abandonné. Les valeurs peuvent être spécifiées en secondes [s] ou millisecondes [ms], avec le défaut étant 40 secondes. Si l'option n'est pas sélectionnée, alors il n'y a pas de limite de temps pour abandonner un transfert.
- *Délai d'expiration de la phase de connexion :* Il s'agit de la limite de temps au sein de laquelle la connexion doit être établie, y compris le temps pris pour les négociations de sécurité. Les valeurs peuvent être spécifiées en secondes [s] ou millisecondes [ms], avec le défaut étant 300 secondes. Ce délai d'expiration ne peut être désactivé :

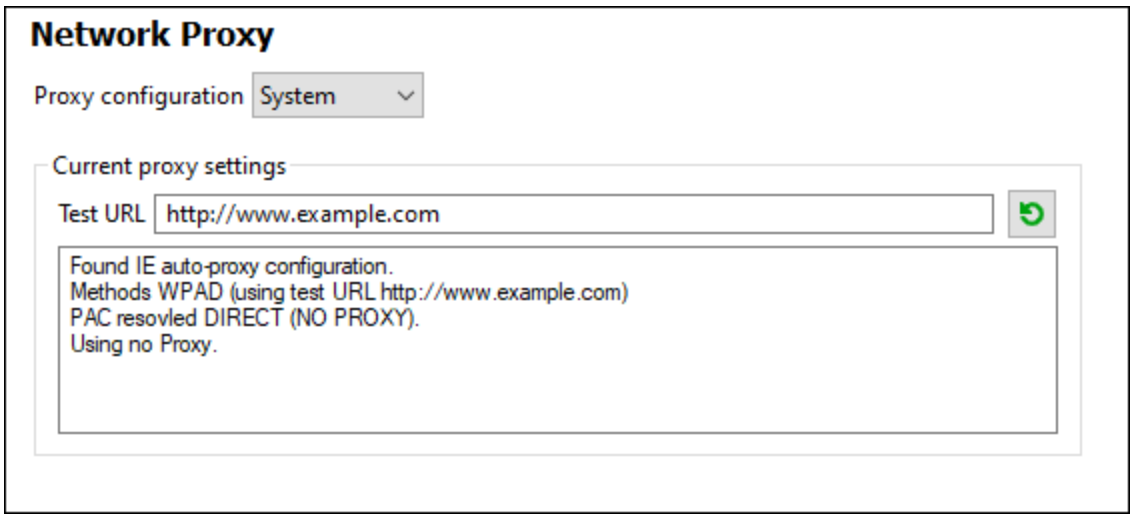
Certificat
- *Vérifier le certificat de serveur TLS/SSL :* Si sélectionné, alors l'authenticité du certificat de serveur est vérifiée en contrôlant la chaîne de signatures numériques jusqu'à ce qu'un certificat racine de confiance est atteint. Cette option est activée par défaut. Si cette option n'est pas sélectionnée, alors la communication n'est pas sécurisée, et les attaques (par exemple, une attaque « man-in-the-middle ») ne serait pas détectée. Notez que cette option ne vérifie pas que le certificat est en fait pour le serveur avec lequel il communique. Pour permettre une sécurité entière, les deux certificats et l'identité doivent être vérifiés (*voir la prochaine option*).
- *Vérifier l'identité de serveur TLS/SSL :* Si sélectionné, alors le certificat de serveur est vérifié pour appartenir au serveur avec lequel nous voulons communiquer. Ceci est fait en vérifiant que le nom de

serveur dans l'URL est le même que le nom dans le certificat. Cette option est activée par défaut. Si cette option n'est pas sélectionnée, alors l'identité du serveur n#est pas vérifiée. Notez que cette option n'active pas la vérification du certificat du serveur. 'Pour permettre une sécurité entière, les deux certificats et l'identité doivent être vérifiés (*voir l'option précédente*).

11.10.3 Proxy de réseau

La section *Proxy de réseau* vous permet de configurer des paramètres de proxy personnalisés. Ces paramètres ont une incidence sur la manière dont l'application se connecte à Internet (par exemple, pour une validation XML). Par défaut, l'application utilise les paramètres proxy du système, dans la plupart des cas, vous n'aurez donc pas à changer les paramètres de proxy. Si vous souhaitez, toutefois, définir un proxy réseau alternatif, dans la zone de liste de *configuration Proxy*, sélectionnez *Automatic* ou *Manual* pour configurer les paramètres en conséquence.

Note : les paramètres de proxy de réseau sont partagés parmi toutes les applications de Altova MissionKit. Donc, si vous modifiez les paramètres dans une application, toutes les applications de MissionKit en seront touchées.

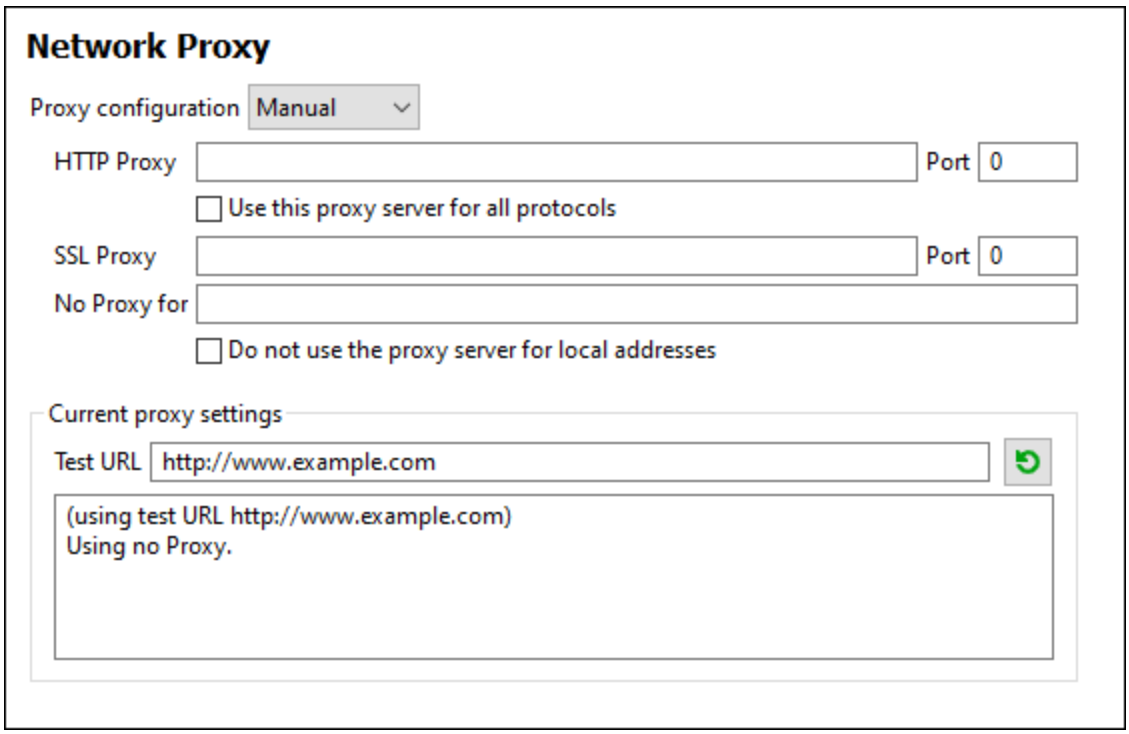


Utiliser les paramètres de proxy de système
Utilise les paramètres Internet Explorer (IE) configurables par le biais des paramètres de proxy du système. Effectue également une requête des paramètres configurés avec **`netsh.exe winhttp`**.

Configuration de proxy automatique
Les options suivantes sont possibles :

- *Paramètres auto-détection :* Consulte un script WPAD (**`http://wpad.LOCALDOMAIN/wpad.dat`**) par le biais de DHCP ou DNS, et utilise ce script pour une configuration proxy.
- *URL de script :* Spécifie une URL HTTP dans un script configuration-auto-proxy (**`.pac`**) qui doit être utilisé pour cette configuration de proxy.
- *Recharger :* Réinitialise et recharge la configuration automatique de proxy actuelle. Cette action requiert Windows 8 ou plus, et peut prendre jusqu'à 30 sec avant de prendre effet.

Configuration de proxy manuelle
Spécifier manuellement le nom d'hôte et le port entièrement qualifiés pour les proxies des protocoles respectifs. Un scheme pris en charge peut être inclus dans le nom d'hôte (par exemple : **`http://hostname`**). Il n'est pas exigé que le schéma soit le même que le protocole respectif si le proxy prend en charge le schéma.



Les options suivantes sont possibles :

- *Proxy HTTP :* Utilise le nom d'hôte spécifié et le port pour le protocole HTTP. Si *Utiliser le serveur proxy pour tous les protocoles* est sélectionné, alors le proxy HTTP spécifié est utilisé pour tous les protocoles.
- *Proxy SSL :* Utilise le nom d'hôte spécifié et le port pour le protocole SSL.
- *Pas de proxy pour :* Une liste séparée par point-virgule (`;`) de nom d'hôtes entièrement qualifiés, de noms de domaine, ou d'adresses IP pour des hôtes qui doivent être utilisés sans un proxy. Les adresses IP ne doivent pas être abrégées et les adresses IPv6 doivent être entourées de crochets (par exemple : **`[2606:2800:220:1:248:1893:25c8:1946]`**). Les noms de domaine doivent commencer avec un point (par exemple : **`.example.com`**).
- *Ne pas utiliser le serveur proxy pour les adresses locales :* Si cochées, ajoute **`<local>`** à la liste *Pas de proxy pour*. Si cette option est sélectionnée, les éléments suivants n'utiliseront pas le proxy : (i) **`127.0.0.1`**, (ii) **`[::1]`**, (iii) tous les noms d'hôte ne contenant pas de point (`.`).

Note : Si un serveur proxy a été défini et que vous voulez déployer une transformation sur Altova FlowForce Server, vous devez sélectionner l'option *Ne pas utiliser le serveur de proxy pour les adresses locales*.

Paramètres de proxy actuels
Fournit un journal verbeux de la détection de proxy. Il peut être réinitialisé avec la touche **Réinitialiser** située à droite du champ *Tester URL* (par exemple, en changeant l'URL de test, ou lorsque les paramètres de proxy ont été modifiés).

- *URL test :* Une URL test peut être utilisée pour voir quel proxy est utilisé pour cette URL spécifique. Aucun E/S n'est effectué avec cette URL. Ce champ ne doit pas être vide si configuration-auto-proxy est utilisé (soit par le biais de *Utiliser paramètres de proxy de système* soit *Configuration proxy automatique*).

11.11 Fenêtre

Cette rubrique recense toutes les commandes de menu disponibles dans le menu **Fenêtre**.

- Cascade

  Cette commande réarrange toutes les fenêtres de document ouvertes de manière à ce qu'elles se présentent en cascade (donc, étalées) l'une sur l'autre.

- Mosaïque horizontale

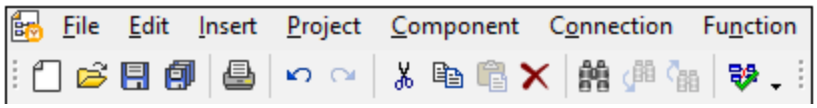
  Cette commande réarrange toutes les fenêtres de document ouvertes sous forme de mosaïque horizontale, les rendant toutes visibles en même temps.

- Mosaïque verticale

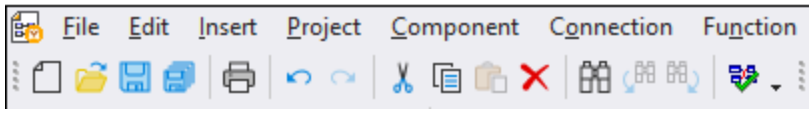
  Cette commande réarrange toutes les fenêtres de document ouvertes mosaïque verticale, les rendant toutes visibles en même temps.

- Thème classique/clair/sombre

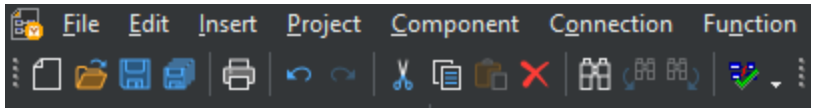
  MapForce vous permet de choisir des thèmes différents : *Classique*, *Clair* et *Sombre*. Les exemples de ces thèmes sont illustrés dans la capture d'écran ci-dessous. L'option par défaut est le thème classique.



Thème classique



Thème clair



Thème sombre

- 1 <MappingName>

  Se réfère au premier design de mappage ouvert. S'il y a plus de mappages ouverts en même temps, ils seront également recensés dans le menu contextuel.

- Windows

  Cette liste affiche toutes les fenêtres actuellement ouvertes et vous permet de basculer rapidement entre elles. Vous pouvez utiliser les raccourcis de clavier **Ctrl-TAB** ou **CTRL F6** pour basculer entre les fenêtres ouvertes.

11.12 Aide

Cette rubrique recense toutes les commandes de menu disponibles dans le menu **Aide**.

- Aide (F1)

  La commande **Aide (F1)** ouvre la documentation d'Aide de l'application (son manuel utilisateur). Par défaut, l'Aide en ligne sous le format HTML sur le site web d'Altova sera ouvert.

  Si vous n'avez pas d'accès Internet ou ne voulez pas, pour une raison ou une autre, accédez à l'Aide en ligne, vous pouvez utiliser la version du manuel utilisateur stockée localement. La version locale est un fichier PDF appelé `MapForce.pdf` qui est stocké dans le dossier d'application (dans le dossier des fichiers de programme).

  Si vous voulez changer le format par défaut pour ouvrir (Aide en ligne ou PDF local), faites-le dans la section Aide du dialogue des Options (commande de menu **Outils | Options**).

- Activation logiciel

  Mettre sous licence votre produit
  Après avoir téléchargé votre logiciel de produits Altova, vous pourrez acquérir une licence - ou l'activer - en utilisant soit une clé d'évaluation gratuite ou en achetant une clé de licence permanente.

  - ***Licence d'évaluation gratuite.*** Lorsque vous lancez le logiciel pour la première fois après l'avoir téléchargé et installé, le dialogue **Activation du logiciel** s'ouvrira. Vous y trouverez un bouton pour demander une licence d'évaluation gratuite. Cliquez dessus pour obtenir votre licence. Quand vous cliquez sur ce bouton, votre ID de l'appareil sera hashé et envoyé à Altova via HTTPS. L'information liée à la licence sera envoyée à l'appareil via une réponse HTTP. Si la licence est créée avec succès, un dialogue à cet effet apparaîtra dans votre application d'Altova. En cliquant **OK** dans ce dialogue, le logiciel sera activé pour une période de 30 jours **sur cet appareil particulier**.

  - ***Clé de licence permanente.*** Le dialogue **Activation du logiciel** contient un bouton pour acheter une clé de licence permanente. Cliquer sur ce bouton pour vous rendre à la boutique en ligne d'Altova, où vous pourrez acheter une clé de licence permanente pour votre produit. Votre licence vous sera envoyée par e-mail sous forme d'un fichier de licence contenant vos données de licence.

    Il existe trois types de licences permanentes : *installée*, *utilisateur simultané*, et *utilisateur nommé*. Une licence installée déverrouille le logiciel sur un seul ordinateur. Si vous achetez une licence installée pour*N* ordinateurs, la licence permettra une utilisation du logiciel sur jusqu'à *N* ordinateurs. Une licence utilisateur concomitant pour *N* utilisateurs concomitants permet à *N* utilisateurs d'exécuter le logiciel simultanément. (Le logiciel peut être installé sur 10N ordinateurs.) Une licence utilisateur nommé autorise un utilisateur spécifique d'utiliser le logiciel sur jusqu'à 5 ordinateurs différents. Pour activer votre logiciel, cliquer sur **Charger une Nouvelle licence**, et, dans le dialogue qui apparaît, chercher ou saisir le chemin vers le fichier de licence et cliquer sur **OK**.

  Note : en ce qui concerne les licences utilisateurs multiples, chaque utilisateur sera invité à saisir son nom dans le champ Nom.

Votre e-mail de licence et les différents moyens de mise sous licence de votre produit Altova (activation) à votre disposition
L'e-mail de licence que vous avez reçu de la part d'Altova contiendra votre fichier de licence en pièce jointe. Le fichier de licence a une extension de fichier **.altova_licenses**.

Pour activer votre produit Altova, vous pouvez choisir une des étapes suivantes :

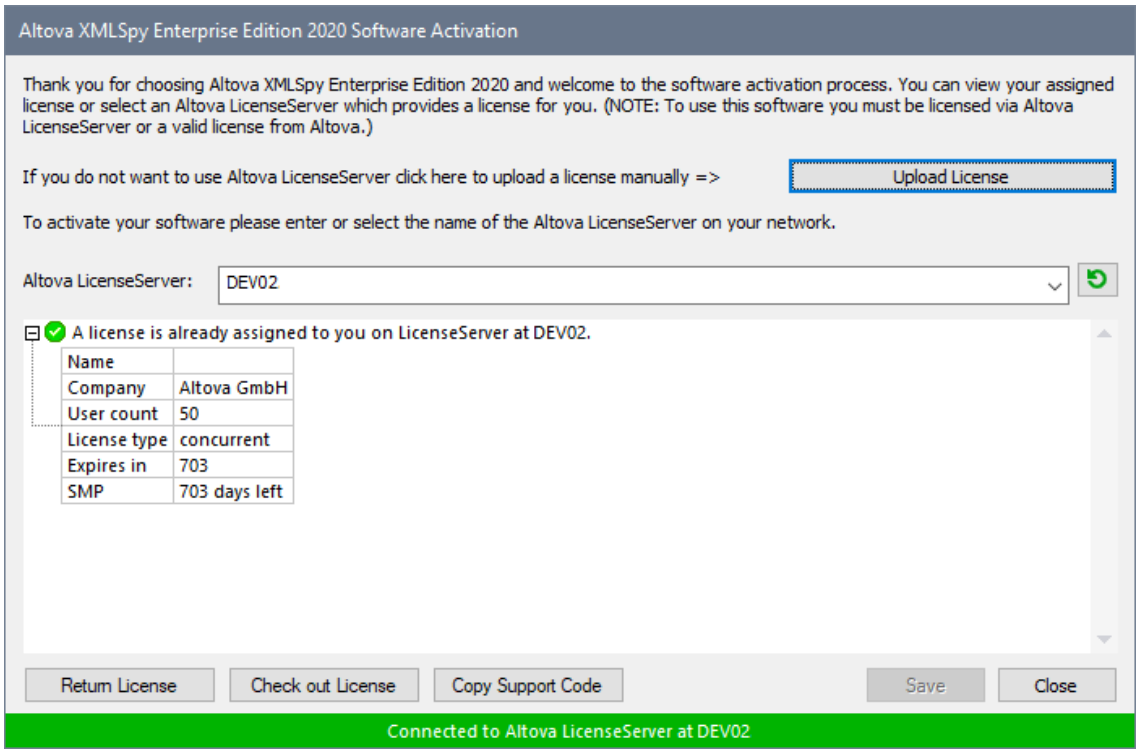
- Enregistrez le fchier de licence (**.altova_licenses**) vers un emplacement approprié, double-cliquez sur le fichier de licence, saisissez tout détail requis dans le dialogue qui apparaît, et terminez en cliquant sur **Appliquer clés**.
- Enregistrez le fichier de licence (**.altova_licenses**) vers un emplacement approprié. Dans votre produit Altova, sélectionnez la commande de menu **Aide | Activation Logiciel**, puis **Télécharger une nouvelle licence**. Chercher ou saisir le champ vers le fichier de licence, et cliquer sur **OK**.
- Enregistrez le fichier de licence (**.altova_licenses**) vers un emplacement approprié et chargez-le depuis cet emplacement vers le pool de licences de votre Altova LicenseServer Vous pouvez ensuite : (i) acquérir la licence depuis votre produit Altova via le dialogue d'Activation du logiciel du produit *(voir ci-dessous)* ou (ii) attribuer la licence au produit depuis l'Altova LicenseServer. *Pour plus d'informations concernant la mise sous licence via LicenseServer, lire le reste de cette rubrique.*

Le dialogue **Activation du logiciel** (*capture d'écran ci-dessous*) peut être accédé à tout moment en cliquant sur la commande **Aide | Activation du logiciel**.

Activer votre logiciel
Vous pouvez activer le logiciel en inscrivant le dialogue Activation du logiciel ou en enregistrant la licence par le biais du Serveur de licence Altova (*voir les détails ci-dessous*).

- *Enregistrant la licence dans le dialogue d'Activation du logiciel*. Dans le dialogue, cliquez sur **Charger une Nouvelle Licence**, puis cherchez et sélectionnez le fichier de licence. Cliquez sur **OK** pour confirmer le chemin vers le fichier de licence et confirmez toutes les données que vous avez saisies (votre nom dans le cas de licences multi-utilisateur). Terminez en cliquant **Enregistrer**.

- *Mise sous licence par le biais du Serveur de licence Altova sur votre réseau :* Pour acquérir une licence par le biais d'un Altova LicenseServer sur votre réseau, cliquez sur **Utiliser Altova LicenseServer**, situé en bas du dialogue **Activation du logiciel**. Choisissez l'appareil sur lequel le LicenseServer que vous souhaitez utiliser a été installé. Veuillez noter que l'auto-découverte des License Servers fonctionne par le biais d'une diffusion envoyée sur le LAN. Puisque les diffusions sont limitées à un sous-réseau, License Server doit se trouver sur le même sous-réseau que l'appareil client pour la découverte automatique afin de fonctionner. Si elle ne fonctionne pas, saisir le nom du serveur. L'Altova LicenseServer doit disposer d'une licence pour votre produit Altova dans son pool de licence. Si une licence est disponible dans le pool de LicenseServer, cela sera indiqué dans le dialogue d'**Activation du logiciel** (*la capture d'écran ci-dessous affiche le dialogue dans Altova XMLSpy*). Cliquez sur **Enregistrer** pour acquérir la licence.



Une fois qu'une licence spécifique aux appareils (aka installée) a été acquise depuis LicenseServer, elle ne peut pas être retournée au LicenseServer pour une période de sept jours. Après cette période, vous pouvez rendre la licence installée (cliquer sur **Retourner licence**) de manière à ce que la licence puisse être acquise depuis LicenseServer par un autre client. (Néanmoins, un administrateur de LicenseServer peut annuler l'attribution à tout moment d'une licence acquise par le biais de la Web UI du LicenseServer). Veuillez noter qu'un renvoi de la licence n'est applicable qu'aux seules licences sur appareil installées, pas aux licences concurrentes.

Extraire la licence

Vous pouvez consulter une licence du pool de licence pour une période de jusqu'à 30 jours pour que la licence puisse être stockée sur l'appareil de produit. Cela vous permet de travailler hors ligne, ce qui peut être utile, par exemple, si vous souhaitez travailler dans un environnement où vous ne pourrez pas accéder à votre Altova LicenseServer (par exemple, si votre produit Altova est installé sur un ordinateur portable et que vous vous trouvez en déplacement). Tant que la licence est extraite, LicenseServer affiche la licence comme étant utilisée ; elle ne peut donc pas être utilisée par une autre machine. La licence passe automatiquement à l'état d'archivage lorsque la période d'extraction expire. En alternative, une licence extraite peut être archivée à tout moment par le biais du bouton **Archiver** du dialogue d'**Activation du logiciel**.

Pour extraire une licence, procédez comme suit : (i) dans le dialogue d'A**ctivation du logiciel**, cliquez sur **Extraire licence** (*voir la capture d'écran ci-dessus*); (ii) dans le dialogue d'**extraction de la licence** qui apparaît, sélectionnez la période d'extraction que vous souhaitez et cliquez sur **Extraire**. La licence sera extraite. Après avoir extrait la licence, deux choses se produisent : (i) Le dialogue d'**Activation du logiciel** affichera les informations d'extraction, y compris l'heure à laquelle l'extraction expirera, (ii) le bouton **Extraire licence** dans le dialogue se transforme en un bouton **Archiver**. Vous pouvez archiver la licence à nouveau à tout moment en cliquant sur

Archiver. Étant donné que la licence passe automatiquement au statut Archiver à l'issue de la période d'extraction, assurez-vous que la période d'extraction que vous avez choisie couvre bien la période pendant laquelle vous travaillerez hors ligne.

Si la licence étant extraite est une licence Utilisateur Installée ou licence Utilisateur Concurrent, alors la licence est extraite vers l'appareil et disponible à l'utilisateur qui a extrait la licence. Si la licence extraite de est uneLicence Utilisateur Nommée, alors la licence est extraite dans le compte Windows de l'utilisateur nommé. Le check-out de licence fonctionnera pour les appareils virtuels, mais pas pour le desktop virtuel (dans un VDI). Notez que quandune licence Utilisateur Nommée est extraite, les données à identifier ce check-out de licence sont stockées dans le profil utilisateur. Pour que le check-out de licence fonctionne, le profil d'utilisateur doit être stocké sur un appareil local qui sera utilisé pour le travail hors ligne. Si le profil de l'utilisateur est stocké à un emplacement non-local (tel que « file-share »), alors le checkout sera rapporté comme invalide quand l'utilisateur tente de démarrer l'application Altova.

Les check-in de licence doivent être de la même version majeure du produit d'Altova pour lequel la licence a été extraite. Donc assurez-vous d'archiver une licence avant que vous ne mettiez à jour votre produit d'Altova à la prochaine version majeure.

Note : afin de pouvoir effectuer des extractions de licence, la fonction d'extraction doit être activée sur le LicenseServer. Si la fonction n'a pas été activée, vous recevrez un message d'erreur à cet effet lorsque vous essayez de faire le « check out ». Dans ce cas, veuillez contacter votre administrateur de LicenseServer.

Copier code de support
Cliquer sur **Copier code de support** pour copier des détails de licence dans le presse-papiers. Il s'agit des données que vous devrez fournir en cas de demande d'assistance avec le formulaire d'assistance en ligne.

Altova LicenseServer offre aux administrateurs IT un aperçu en temps réel de toutes les licences Altova sur un réseau, avec les détails de chaque licence, ainsi que les attributions clients et l'utilisation client des licences. L'avantage d'utiliser LicenseServer réside donc dans les fonctions administratives qu'il offre pour la gestion de licence à large volume d'Altova. Altova LicenseServer est disponible gratuitement depuis le site web Altova. Pour plus d'informations concernant Altova LicenseServer et la mise sous licence par le biais d'Altova LicenseServer, voir la documentation Altova LicenseServer.

Formulaire de commande

Lorsque vous êtes prêt pour commander une version de licence du produit de logiciel, vous pouvez soit utiliser la touche **Acheter une clé de licence permanente** dans le dialogue **Activation du logiciel** (*voir la section précédente*) ou la commande **Formulaire de commande** pour continuer vers la boutique en ligne Altova sécurisée.

Inscription

Ouvre la page d'enregistrement du produit Altova dans un onglet de votre navigateur. L'enregistrement de votre logiciel Altova vous aidera à vous assurer de toujours rester à jour avec les dernières informations du produit.

Vérifier les mises à jour

Contrôle sur le serveur Altova si une version plus récente que la vôtre est actuellement disponible et, dans l'affirmative, affiche un message approprié.

- Centre de support

  Un lien qui vous mènera vers le Centre de support Altova sur Internet. Le Centre de support contient des FAQ, des forums de discussion pour toute sorte de problèmes et l'accès à l'équipe de support technique d'Altova.

- Télécharger les composants et les outils gratuits

  Un lien menant au Centre de téléchargement des composants Altova sur Internet. À partir de là, vous pouvez télécharger une variété de logiciels complémentaires à utiliser avec des produits Altova. Ces logiciels vont de processeurs XSLT et XSL-FO à des Plateformes de serveur d'application. Les logiciels disponibles dans le Centre de téléchargement des composants sont généralement gratuits.

- MapForce sur Internet

  Un lien menant au site web Altova sur Internet. Vous pouvez en apprendre plus sur MapForce, les technologies et produits liés le le site web Altova.

- Formation MapForce

  Un lien menant à la page de Formation en ligne sur le site web Altova. Ici, vous pouvez choisir parmi une série de cours en ligne tenus par des formateurs experts Altova.

- À propos de MapForce

  Affiche la fenêtre d'accueil et le numéro de version de votre produit. Si vous utilisez la version 64-bit de MapForce, cela est indiqué par le suffixe (x64) placé après le nom de l'application. Il n'y a pas de suffixe pour la version 32-bit.

12 Annexes

Ces annexes contiennent une information technique sur MapForce, ses aspects techniques et la licence. Elle fournit également une liste de termes clés spécifiques à MapForce et aux produits y afférents. La section est organisée en sous-sections comme suit :

12.1 Notes de prise en charge

MapForce® est une application Windows de 32/64-bit exécutée sur les systèmes d'exploitation suivants :

- Windows 10, Windows 11
- Windows Server 2016 ou plus récent

La prise en charge 64-bit est disponible pour les éditions Enterprise et Professional.

12.1.1 Sources et cibles prises en charge

Lorsque vous modifiez le langage de transformation d'un mappage MapForce, certaines fonctions peuvent ne pas être prises en charge par ce langage spécifique. La table suivante résume la compatibilité de formats de mappage et langages de transformation dans **MapForce Basic Edition**.

Remarques :

- *Built-in* signifie que vous pouvez exécuter le mappage en cliquant sur l'onglet **Sortie** dans MapForce, ou l'exécuter avec MapForce Server.

12.1.2 Fonctions prises en charge dans le code généré

La table suivante recense les fonctions pertinentes pour la génération de code et l'étendue de la prise en charge dans chaque langage dans **MapForce Basic Edition**.

Fonction	XSLT 1.0	XSLT 2.0	XSLT 3.0
Fournir des paramètres au mappage 147	●	●	●
Fournir les noms de fichier d'entrée dynamiquement depuis le mappage 400	●	●	●
Fournir des noms de fichier de caractère générique en tant qu'entrée de mappage 400 1		●	●
Générer les noms de fichier de sortie dynamiquement depuis le mappage 400		●	●
Retourner des valeurs de string depuis le mappage 154	●	●	●
Variables 158		●	●
Composants de tri 170		●	●
Fonctions de regroupement 276		●	●
Filtres 176	●	●	●

Fonction	XSLT 1.0	XSLT 2.0	XSLT 3.0
Composants Value-Map 182	●	●	●
Noms de nœuds dynamiques 383		●	●

Notes de bas de page :

1. XSLT 2.0, XSLT 3.0 et XQuery utilisent la fonction **`fn:collection`**. La mise en place dans les moteurs Altova XSLT 2.0, XSLT 3.0 et XQuery résolvent des caractères génériques. D'autres moteurs peuvent se comporter différemment.

12.2 Information des moteurs

Cette section contient des informations concernant les fonctions spécifiques à la mise en place du Valideur XML Altova XML, Altova XSLT 1.0 Engine, Altova XSLT 2.0 Engine et Altova XQuery Engine.

12.2.1 Informations concernant le moteur XSLT et XQuery

Les moteurs XSLT et XQuery de MapForce suivent de près les spécifications W3C et sont donc plus strictes que les moteurs Altova précédents, comme dans les versions précédentes de XMLSpy. Ainsi, de petites erreurs qui étaient ignorées par les moteurs précédents sont maintenant marquées en tant qu'erreurs par MapForce.

Par exemple :

- Il s'agit d'une erreur de type (`err:XPTY0018`) si le résultat d'un opérateur de chemin contient aussi bien les nœuds que les non-nœuds.
- Il s'agit d'une erreur de type (`err:XPTY0019`) si `E1` dans une expression de chemin `E1/E2` n'évalue pas à une séquence de nœuds.

Si vous rencontrez ce type d'erreur, modifiez soit le document XSLT/XQuery, soit le document d'instance selon vos besoins.

Cette section décrit les fonctions spécifiques à la mise en place des moteurs, organisée par spécification :

12.2.1.1 XSLT 1.0

Le moteur XSLT 1.0 de MapForce est conforme aux Recommandations XSLT 1.0 du 16 novembre 1999 et aux Recommandations XPath 1.0 du 16 novembre 1999 du World Wide Web Consortium (W3C's). Veuillez noter les informations suivantes concernant l'implémentation.

Notes concernant l'implémentation

Lorsque l'attribut `method` de `xsl:output` est défini sur HTML, ou si la sortie HTML est sélectionnée par défaut, les caractères spéciaux dans le fichier XML ou XSLT sont insérés dans le document HTML en tant que références de caractère HTML dans la sortie. Par exemple, le caractère U+00A0 (la référence de caractère hexadécimale pour un espace insécable) est inséré dans le code HTML soit en tant que référence de caractère (` ` ou ` `) soit en tant que référence d'entité, ` ` .

12.2.1.2 XSLT 2.0

Cette section :

Conformité

Le moteur XSLT 2.0 de MapForce est conforme aux Recommandations XSLT 2.0 du 23 janvier 2007 et aux Recommandations XPath 2.0 du 14 décembre 2010 du World Wide Web Consortium (W3C's).

Rétrocompatibilité

Le moteur XSLT 2.0 est rétrocompatible. Généralement, la compatibilité rétroactive du moteur XSLT 2.0 entre en jeu si vous utilisez le moteur XSLT 2.0 pour traiter une feuille de style XSLT 1.0 ou une instruction. Veuillez noter qu'il peut y avoir des différences dans les sorties produites par le moteur XSLT 1.0 et la rétrocompatibilité du moteur XSLT 2.0.

Espaces de nom

Votre feuille de style XSLT 2.0 devrait déclarer les espaces de noms suivants afin que vous puissiez utiliser les constructeurs de type et les fonctions disponibles dans XSLT 2.0. Les préfixes indiqués ci-dessous sont utilisés de manière conventionnelle ; vous pourriez utiliser les préfixes alternatifs si vous le souhaitez.

Nom d'espace de nom	Préfixe	Espace de nom URI
Types de schéma XML	`xs:`	`http://www.w3.org/2001/XMLSchema`
Fonctions XPath 2.0	`fn:`	`http://www.w3.org/2005/xpath-functions`

Généralement ces espaces de nom seront déclarés sur l'élément `xsl:stylesheet` ou `xsl:transform`, tel que montré dans la liste suivante :

```
<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>
```

Veuillez noter les points suivants :

- Le moteur XSLT 2.0 utilise l'espace de nom XPath 2.0 et Fonctions XQuery 1.0 (recensées dans la table ci-dessus) en tant que son **espace de noms de fonctions par défaut**. Ainsi, vous pouvez utiliser les fonctions XPath 2.0 et XSLT 2.0 dans votre feuille de style sans aucun préfixe. Si vous déclarez l'espace de nom des fonctions XPath 2.0 dans votre feuille de style avec un préfixe, vous pourrez utiliser en plus le préfixe attribué dans la déclaration.
- Lors de l'utilisation des constructeurs de types et des types provenant de l'espace de nom du Schéma XML, le préfixe utilisé dans la déclaration d'espace de nom doit être utilisé lors de l'appel du constructeur de type (par exemple, `xs:date`).
- Certaines fonctions XPath 2.0 portent le même nom que les types de données du schéma XML. Par exemple, pour les fonctions XPath `fn:string` et `fn:boolean`, il existe des types de données du schéma XML portant le même nom local : `xs:string` et `xs:boolean`. Donc si vous décidez d'utiliser l'expression XPath `string('Hello')`, l'expression évalue en tant que `fn:string('Hello')`et non pas en tant que `xs:string('Hello')`.

Compatibilité avec le schéma

Le moteur XSLT 2.0 est compatible avec le schéma. Vous pouvez ainsi utiliser des types de schéma définis par l'utilisateur et l'instruction `xsl:validate`.

Comportement spécifique à l'implémentation

Ci-dessous, vous trouverez une description de la gestion du moteur XSLT 2.0 des aspects spécifiques à l'implémentation du comportement de certaines fonctions XSLT 2.0.

`xsl:result-document`
Les encodages pris en charge en supplément sont (les codes spécifiques à Altova) : `x-base16tobinary` et `x-base64tobinary`.

`function-available`
La fonction teste la disponibilité des fonctions in-scope (XSLT, XPath, et fonctions d'extension).

`unparsed-text`
L'argument `href` accepte (i) les chemins relatifs pour les fichiers dans le dossier base-uri et (ii) les chemins absolus avec ou sans `file://` protocol. Les encodages pris en charge de manière supplémentaire sont (spécifiques à Altova): `x-binarytobase16` et `x-binarytobase64`. Exemple : `xs:base64Binary(unparsed-text('chart.png', 'x-binarytobase64'))`.

`unparsed-text-available`
L'argument `href` accepte (i) les chemins relatifs pour les fichiers dans le dossier base-uri et (ii) les chemins absolus avec ou sans `file://` protocol. Les encodages pris en charge de manière supplémentaire sont (spécifiques à Altova): `x-binarytobase16` et `x-binarytobase64`.

Note : les valeurs d'encodage suivantes, qui sont mises en œuvre dans des versions antérieures du produit prédécesseur de RaptorXML, AltovaXML, sont dépcréciées à présent : `base16tobinary`, `base64tobinary`, `binarytobase16` and `binarytobase64`.

12.2.1.3 XQuery 1.0

Cette section :

Conformité

Le moteur XQuery 1.0 de MapForce est conforme à la Recommandation XQuery 1.0 du 14 décembre 2010 du World Wide Web Consortium (W3C's). Le standard XQuery accorde un pouvoir discrétionnaire concernant la mise en place de nombreuses fonctions. Ci-dessous, vous trouverez une liste expliquant comment le moteur XQuery 1.0 implémente ces fonctions.

Compatibilité avec le schéma

Le moteur XQuery 1.0 est **schema-aware**.

Encodage

Les encodages de caractères UTF-8 et UTF-16 sont pris en charge.

Espaces de nom

Les URI d'espace de nom suivant et leurs liaisons associées sont prédéfinies.

Nom d'espaces de nom	Préfixe	URI Espace de noms
Types de schéma XML	`xs:`	`http://www.w3.org/2001/XMLSchema`
Instance de schéma	`xsi:`	`http://www.w3.org/2001/XMLSchema-instance`
Fonctions intégrées	`fn:`	`http://www.w3.org/2005/xpath-functions`
Fonctions locales	`local:`	`http://www.w3.org/2005/xquery-local-functions`

Veuillez noter les points suivants :

- Le moteur XQuery 1.0 Engine reconnaît les préfixes recensés ci-dessus comme étant liés aux espaces de noms correspondants.
- Étant donné que l'espace de noms des fonctions intégrées recensé ci-dessus est l'espace de noms des fonctions par défaut dans XQuery, le préfixe `fn:` ne doit pas nécessairement être utilisé lorsque des fonctions intégrées sont invoquées (par exemple, `string("Hello")` appellera la fonction `fn:string`). Néanmoins, le préfixe `fn:` peut être utilisé pour appeler une fonction intégrée sans avoir à déclarer l'espace de noms dans le prologue query (par exemple : `fn:string("Hello")`).
- Vous pouvez changer l'espace de noms des fonctions par défaut en déclarant l'expression `default function namespace` dans le prologue de de requête.
- En cas d'utilisation des types depuis l'espace de noms du Schéma XML, le préfixe `xs:` peut être utilisé sans devoir déclarer explicitement les espace de noms et lier ces préfixes dans le prologue de requête. (Exemple : `xs:date` and `xs:yearMonthDuration`.) Si vous souhaitez utiliser d'autres préfixes pour l'espace de noms du schéma XML, cela doit être déclaré explicitement dans le prologue de requête. (Exemple: `declare namespace alt = "http://www.w3.org/2001/XMLSchema"; alt:date("2004-10-04")`.)
- Veuillez noter que les types de données `untypedAtomic`, `dayTimeDuration`, et `yearMonthDuration` ont été déplacés, avec les CR de 23 January 2007, depuis l'espace de noms des Types de données XPath vers l'espace de noms du schéma XML, donc : `xs:yearMonthDuration`.

Si des espaces de noms pour les fonctions, les constructeurs de type, les tests de nœud, etc. sont mal attribués, une erreur est rapportée. Veuillez noter, néanmoins, que certaines fonctions portent le même nom que les types de données de schéma, par ex. `fn:string` et `fn:boolean`. (Les deux `xs:string` et `xs:boolean` sont définis.) Le préfixe d'espace de noms détermine si la fonction ou le constructeur de type est utilisé.

XML document de source et validation

Les documents XML utilisés dans l'exécution d'un document XQuery avec le moteur XQuery 1.0 doit être bien formé. Néanmoins, ils ne doivent pas être valides conformément à un schéma XML. Si le fichier n'est pas valide, le fichier invalide est chargé sans information de schéma. Si le fichier XML est associé avec un schéma interne et est valide conformément à ce schéma, l'information de validation post-schéma sera générée pour les données XML et sera utilisée pour l'évaluation de requête.

Contrôle de type statique et dynamique

La phase d'analyse statique contrôle les aspects de la requête comme la syntaxe, si des références externes existent (par ex. pour les modules), si des fonctions et des variables invoquées sont définies, etc. Si une erreur est détectée dans la phase de l'analyse statique, elle sera rapportée et l'exécution sera stoppée.

Le contrôle de type dynamique est effectué lors de l'exécution, lorsque la requête est réellement exécutée. Si un type est incompatible avec les exigences d'une opération, une erreur sera rapportée. Par exemple, l'expression `xs:string("1") + 1` retourne une erreur parce que l'opération d'édition ne peut pas être effectuée sur un opérande de type `xs:string`.

Modules de bibliothèque

Les modules de Bibliothèque stockent les fonctions et les variables de manière à ce qu'elles puissent être réutilisées. Le moteur XQuery 1.0 prend en charge des modules qui sont stockés **dans un seul fichier XQuery externe**. Un tel fichier de module doit contenir une déclaration `module` dans son prologue, qui associe un espace de noms cible. Voici un module d'exemple :

```
module namespace libns="urn:module-library";
declare variable $libns:company := "Altova";
declare function libns:webaddress() { "http://www.altova.com" };
```

Toutes les fonctions et les variables déclarées dans le module font partie de l'espace de noms associé au module. Celui-ci est utilisé en l'important dans un fichier XQuery avec l'instruction `import module` se trouvant dans le prologue de requête. L'instruction `import module` importe uniquement les fonctions et les variables déclarées directement dans le fichier de module de bibliothèque. Comme suit :

```
import module namespace modlib = "urn:module-library" at "modulefilename.xq";
if        ($modlib:company = "Altova")
then       modlib:webaddress()
else       error("No match found.")
```

Fonctions externes

Les fonctions externes ne sont pas prises en charge, c.à.d. dans les expressions utilisant le mot-clé `external`, comme dans :

```
declare function hoo($param as xs:integer) as xs:string external;
```

Collations

La collation par défaut est la collation de point de code Unicode, qui compare les chaînes sur la base de leur point de code Unicode. Les autres collations prises en charge sont les collations ICU recensées ici[492]. Pour

utiliser une collation spécifique, fournir son URI tel que donné dans la liste des collations prises en charge 492. Toute comparaison de chaîne, y compris pour les fonctions `fn:max` et `fn:min` seront effectuées conformément à la collation spécifiée. Si l'option de collation n'est pas spécifiée, la collation de point de code Unicode par défaut est utilisée.

Précision des types numériques

- Le type de données `xs:integer` est une précision arbitraire, c.à.d. il peut représenter n'importe quel chiffre.
- Le type de données `xs:decimal` a une limite de 20 chiffres après la virgule.
- Les types de données `xs:float` et `xs:double` ont une précision limitée de 15 chiffres.

Prise en charge des instructions XQuery

L'instruction `Pragma` n'est pas prise en charge. Si elle survient, elle sera ignorée et l'expression de fallback sera évaluée.

Comportement spécifique à la mise en œuvre

CI-dessous, vous trouverez une description pour savoir comment les moteurs XQuery et XQuery Update 1.0 gèrent les aspects de certaines fonctions spécifiques à la mise en œuvre.

`unparsed-text`
L'argument `href` accepte (i) les chemins relatifs pour les fichiers dans le dossier base-uri et (ii) les chemins absolus avec ou sans `file://` protocol. Les encodages pris en charge de manière supplémentaire sont (spécifiques à Altova): `x-binarytobase16` et `x-binarytobase64`. Exemple : `xs:base64Binary(unparsed-text('chart.png', 'x-binarytobase64'))`.

`unparsed-text-available`
L'argument `href` accepte (i) les chemins relatifs pour les fichiers dans le dossier base-uri et (ii) les chemins absolus avec ou sans `file://` protocol. Les encodages pris en charge de manière supplémentaire sont (spécifiques à Altova): `x-binarytobase16` et `x-binarytobase64`.

Note : les valeurs d'encodage suivantes, qui sont mises en œuvre dans des versions antérieures du produit prédécesseur de RaptorXML, AltovaXML, sont dépcréciées à présent : `base16tobinary`, `base64tobinary`, `binarytobase16` and `binarytobase64`.

12.2.2 Fonctions XSLT et XPath/XQuery

Cette section réunit les fonctions d'extension Altova et d'autres fonctions d'extension qui peuvent être utilisées dans les expressions XPath et/ou XQuery. Les fonctions d'extension Altova peuvent être utilisées avec les moteurs XSLT et XQuery d'Altova, et elles offrent des fonctions supplémentaires à celles disponibles dans les bibliothèques de fonctions définies dans les standards W3C.

Cette section décrit principalement les fonctions d'extension XPath/XQuery qui ont été créées par Altova pour fournir des opérations supplémentaires. Ces fonctions 494 peuvent être calculées par les moteurs XSLT et XQuery d'Altova selon les règles décrites dans cette section. Pour information sur les fonctions XPath/XQuery régulières, voir la Fonction de référence XPath/XQuery d'Altova.

Points généraux

Les points généraux suivants devraient être notés :

- Les fonctions des bibliothèques de fonction core définies dans les spécifications W3C speuvent être appelées sans préfixe. La raison étant que les moteurs XSLT et XQuery d'Altova lisent les fonctions sans préfixe comme appartenant à l'espace de nom `http://www.w3.org/2005/xpath-functions`, qui est l'espace de nom des fonctions par défaut spécifiées dans les spécifications des fonctions XPath/XQuery. Si cet espace de nom est déclaré explicitement dans un document XSLT ou XQuery, le préfixe utilisé dans la déclaration d'espace de nom peut aussi être utilisé en option sur les noms de fonction.
- En général, si une fonction escompte une séquence d'un item en tant qu'argument, et qu'une séquence de plus d'un item est soumise, une erreur sera retournée.
- Toutes les comparaisons de strings sont réalisées en utilisant la collation de point de code Unicode.
- Les résultats qui sont des QNames sont sérialisés sous la forme `[prefix:]localname`.

Précision de la décimale xs:
La précision se réfère au nombre de chiffres dans le nombre et la spécification requiert un minimum de 18 chiffres. Pour les opérations de division qui produisent un résultat de type `xs:decimal`, la précision est de 19 chiffres après le point décimal sans arrondissement.

Fuseau horaire implicite
Lorsque deux valeurs `date`, `time`, ou `dateTime` doivent être comparées, le fuseau horaire des valeurs comparées doit être connu. Si le fuseau n'est explicitement donné dans une telle valeur, le fuseau horaire implicite est utilisé. Le fuseau horaire implicite est prélevé de l'horloge du système et sa valeur peut être contrôlée avec la fonction `implicit-timezone()`.

Collations
La collation par défaut est la collation de point de code Unicode qui compare les chaînes sur la base de leur point de code Unicode. Le processeur utilise l'Unicode Collation Algorithm. D'autres collations prises en charge sont les collations ICU recensées ci-dessous ; pour en utiliser une, fournissez son URI tel qu'énnoncé dans la table ci-dessous. Toute comparaison de chaîne, y compris en ce qui concerne les fonctions `max` et `min`, sera effectuée conformément à la collation spécifiée. Si l'option de collation n'est pas spécifiée, la collation de point de code Unicode par défaut sera utilisée.

Langage	URI
`da`: Danois	`da_DK`
`de`: Allemand	`de_AT, de_BE, de_CH, de_DE, de_LI, de_LU`
`en`: Anglais	`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: Espagnol	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: Français	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: Italien	it_CH, it_IT
ja: Japonais	ja_JP
nb: Norvégien Bokmål	nb_NO
nl: Néerlandais	nl_AW, nl_BE, nl_NL
nn: Nynorsk	nn_NO
pt: Portugais	pt_AO, pt_BR, pt_GW, pt_MZ, pt_PT, pt_ST
ru: Russe	ru_MD, ru_RU, ru_UA
sv: Suédois	sv_FI, sv_SE

Axe du nom d'espace
L'axe du nom d'espace est devenu obsolète dans XPath 2.0. Néanmoins, l'utilisation de l'axe du nom d'espace est prise en charge. Pour accéder aux information de l'espace de nom, avec des mécanismes XPath 2.0, utilisez les fonctions `in-scope-prefixes()`, `namespace-uri()` et `namespace-uri-for-prefix()`.

12.2.2.1 Fonctions d'extension Altova

Les fonctions d'extension Altova peuvent être utilisées dans les expressions XPath/XQuery. Elles fournissent des fonctions supplémentaires aux fonctions d'ores et déjà disponibles dans la librairie standard des fonctions XPath, XQuery et XSLT. Les fonctions d'extension Altova se trouvent dans **l'espace de nom des fonctions d'extension Altova, `http://www.altova.com/xslt-extensions`**, et sont indiquées dans cette section par le préfixe **`altova:`**, qui est présupposé être lié à cet espace de nom. Veuillez noter que, en ce qui concerne les versions futures de votre produit, la prise en charge d'une fonction peut être interrompue et le comportement de certaines fonctions peut changer. Veuillez consulter la documentation lors des publications à venir pour plus d'informations concernant la prise en charge des fonctions d'extension Altova de cette version.

Les fonctions définies dans les spécifications de Fonctions XPath/XQuery de W3C peuvent être utilisées dans : (i) les expressions XPath dans un contexte XSLT, et (ii) dans les expressions XQuery dans un document XQuery. Dans cette documentation, nous indiquons les fonctions à utiliser dans le contexte précédent (XPath dans XSLT) avec un symbole **`XP`** et les appelons fonctions XPath ; les fonctions qui peuvent être utilisées dans le contexte à venir (XQuery) sont indiquées avec un symbole **`XQ`** ; elles fonctionnent en tant que fonctions XQuery. Les spécifications XSLT de W3C —pas les spécifications de Fonctions XPath/XQuery —définissent également les fonctions qui peuvent être utilisées dans des expressions XPath dans des documents XSLT. Ces fonctions sont marquées avec un symbole **`XSLT`** et sont appelées fonctions XSLT. Les versions XPath/XQuery et XSLT dans lesquelles une fonction peut être utilisée sont indiquées dans la description de la fonction (*voir symboles ci-dessous*). Les fonctions provenant des librairies de fonction XPath/XQuery et XSLT

sont recensées dans un préfixe. Les fonctions d'extension provenant d'autres librairies, comme les fonctions d'extension Altova, sont regroupés avec un préfixe.

Fonctions XPath (utilisées dans les expressions XPath dans XSLT) :	XP1 XP2 XP3.1
Fonctions XSLT (utilisées dans les expressions XPath dans XSLT) :	XSLT1 XSLT2 XSLT3
Fonctions XQuery (utilisées dans les expressions XQuery dans XQuery) :	XQ1 XQ3.1

Usage de Fonctions d'extension Altova

Pour pouvoir utiliser les fonctions d'extension d'Altova, vous devez déclarer l'espace de nom des fonctions d'extension d'Altova (*d'abord mettre en surbrillance dans la liste de codes ci-dessous*) puis utiliser les fonctions d'extension pou qu'elles soient résolues comme appartenant à cet espace de noms (*voir deuxième mise en surbrillance*). L'exemple ci-dessous utilise la fonction d'extension d'Altova appelée **âge**.

```
<xsl:stylesheet version="2.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:fn="http://www.w3.org/2005/xpath-functions"
    xmlns:altova="http://www.altova.com/xslt-extensions">
    <xsl:output method="text" encoding="ISO-8859-1"/>
    <xsl:template match="Persons">
        <xsl:for-each select="Person">
            <xsl:value-of select="concat(Name, ': ')"/>
            <xsl:value-of select="altova:age(xs:date(BirthDate))"/>
            <xsl:value-of select="' years&#x0A;'"/>
        </xsl:for-each>
    </xsl:template>
</xsl:stylesheet>
```

Fonctions XSLT 496

Les fonctions XSLT peuvent uniquement être utilisées dans des expressions XPath dans un contexte XSLT (comme les fonctions `current-group()` ou `key()` de XSLT 2.0). Ces fonctions ne sont pas prévues pour, et ne fonctionneront pas dans un contexte non-XSLT (par exemple, dans un contexte XQuery). Les fonctions XBRL Altova peuvent uniquement être utilisées avec des éditions des produits Altova qui présentent une prise en charge XBRL.

Fonctions XPath/XQuery

Les fonctions XPath/XQuery peuvent être utilisées tous les deux dans les expressions XPath dans les contextes XSLT et dans les expressions XQuery :

- Date/Heure 499
- Géolocalisation 516
- Liée à l'image 525
- Numérique 530

- Séquence 552
- String 561
- Divers 567

12.2.2.1.1 Fonctions XSLT

Les fonctions d'extension XSLT peuvent être utilisées dans les expressions XPath dans un contexte XSLT. Elles ne fonctionneront pas dans un contexte non-XSLT (par exemple dans un contexte XQuery).

Note concernant le nommage de fonctions et de l'applicabilité de la langue

Les fonctions d'extension Altova peuvent être utilisées dans les expressions XPath/XQuery. Elles fournissent des fonctions supplémentaires aux fonctions d'ores et déjà disponibles dans la librairie standard des fonctions XPath, XQuery et XSLT. Les fonctions d'extension Altova se trouvent dans **l'espace de nom des fonctions d'extension Altova, `http://www.altova.com/xslt-extensions`**, et sont indiquées dans cette section par le préfixe **`altova:`**, qui est présupposé être lié à cet espace de nom. Veuillez noter que, en ce qui concerne les versions futures de votre produit, la prise en charge d'une fonction peut être interrompue et le comportement de certaines fonctions peut changer. Veuillez consulter la documentation lors des publications à venir pour plus d'informations concernant la prise en charge des fonctions d'extension Altova de cette version.

Fonctions XPath (utilisées dans les expressions XPath dans XSLT) :	XP1 XP2 XP3.1
Fonctions XSLT (utilisées dans les expressions XPath dans XSLT) :	XSLT1 XSLT2 XSLT3
Fonctions XQuery (utilisées dans les expressions XQuery dans XQuery) :	XQ1 XQ3.1

Fonctions générales

distinct-nodes [altova:]

`altova:distinct-nodes(`*`node()*`***`)`** **`asnode()*`** XSLT1 XSLT2 XSLT3
Prend un ensemble d'un ou de plusieurs nœuds en tant que son entrée et retourne le même ensemble moins les nœuds avec des valeurs dupliquées. La comparaison s'effectue en utilisant la fonction XPath/XQuery `fn:deep-equal`.

Exemples

- **`altova:distinct-nodes`**`(country)` retourne tous les nœuds `country` enfant moins ceux possédant des valeurs dupliquées.

evaluate [altova:]

`altova:evaluate(XPathExpression` *`as xs:string`*`[, ValueOf$p1, ... ValueOf$pN])` XSLT1 XSLT2 XSLT3
Prend une expression XPath, passée en tant que chaîne, en tant que son argument obligatoire. Elle retourne la sortie de l'expression évaluée. Par exemple : **`altova:evaluate('//Name[1]')`** retourne les contenues du premier élément `Name` dans le document. Veuillez noter que l'expression `//Name[1]` est passée en tant que chaîne en l'enfermant dans des guillemets simples.

La fonction altova:evaluate peut prendre des arguments supplémentaires en option. Ces arguments sont les valeurs des variables in-scope qui portent les noms p1, p2, p3... pN. Veuillez noter les points suivants concernant l'utilisation : (i) Les variables doivent être définies avec les noms de la formule pX, lorsque X est un entier ; (ii) les arguments de la fonction altova:evaluate (*voir signature ci-dessus*), à partir du deuxième argument, fournissent les valeurs de la variables, avec la séquence des arguments correspondant à la séquence des variables classées numériquement : p1 à pN: le deuxième argument sera la valeur de la variable p1, le troisième argument celui de la variable p2, etc. ; (iii) Les valeurs de variable doivent être de type item*.

Exemple

```
<xsl:variable name="xpath" select="'$p3, $p2, $p1'" />
<xsl:value-of select="altova:evaluate($xpath, 10, 20, 'hi')" />
```

outputs "hi 20 10"

Dans les listes ci-dessus, veuillez noter les points suivants :

- Le deuxième argument de l'expression altova:evaluate est la valeur attribuée à la variable $p1, le troisième argument est celui attribué à la variable $p2, etc.
- Veuillez noter que le quatrième argument de la fonction est une valeur de chaîne, ce qui est indiqué par le fait qu'elle est contenue dans des guillemets.
- L'attribut select de l'élément xs:variable fournit l'expression XPath. Puisque cette expression doit être de type xs:string, elle est contenue dans des guillemets simples.

Exemples pour mieux illustrer l'utilisation des variables

- ```
  <xsl:variable name="xpath" select="'$p1'" />
  <xsl:value-of select="altova:evaluate($xpath, //Name[1])" />
  ```
  Sort la valeur du premier élément Name*.*

- ```
  <xsl:variable name="xpath" select="'$p1'" />
  <xsl:value-of select="altova:evaluate($xpath, '//Name[1]')" />
  ```
  Sort "//Name[1]"

La fonction d'extension altova:evaluate() est utile lorsqu'une expression XPath dans la feuille de style XSLT contient une ou plusieurs parties qui doivent être évaluées dynamiquement. Par exemple, prenez comme exemple une situation dans laquelle un utilisateur saisit sa requête pour le critère de tri et le critère est stocké dans l'attribut UserReq/@sortkey. Dans la feuille de style, vous pouvez ensuite avoir l'expression : <xsl:sort select="**altova:evaluate(../UserReq/@sortkey)**" order="ascending"/>. La fonction altova:evaluate() lit l'attribut sortkey de l'élément enfant UserReq du parent du nœud contextuel. Si, par exemple, la valeur de l'attribut sortkey est Price, alors Price est retourné par la fonction altova:evaluate() et devient la valeur de l'attribut select : <xsl:sort select="**Price**" order="ascending"/>. Si cette instruction sort apparaît dans le contexte d'un élément appelé Order, alors les éléments Order seront triés conformément aux valeurs de leurs enfants Price. En alternative, si la valeur de @sortkey était, par exemple, Date, alors les éléments Order seraient triés selon les valeurs de leurs enfants Date. Donc le critère de triage pour Order est choisi à partir de l'attribut sortkey lors de l'exécution. Cela n'aurait pas pu se réaliser avec une expression telle que : <xsl:sort select="**../UserReq/@sortkey**" order="ascending"/>. Dans le cas montré ci-dessus, le critère de tri aurait été l'attribut sortkey lui-même, et non pas Price ou Date (ou tout autre contenu actuel de sortkey).

Note : Le contexte statique inclut des espaces de nom, des types et des fonctions, mais pas des variables, depuis l'environnement d'appel. L'URI de base et l'espace de nom par défaut sont hérités.

Plus d'exemples

- Variables statiques : `<xsl:value-of select="$i3, $i2, $i1" />`
  Sort les valeurs des trois variables.

- Expression XPath dynamique avec des variables dynamiques :
  ```
  <xsl:variable name="xpath" select="'$p3, $p2, $p1'" />
  <xsl:value-of select="altova:evaluate($xpath, 10, 20, 30)" />
  ```
  Sortie "30 20 10"

- Expression XPath dynamique sans variable dynamique :
  ```
  <xsl:variable name="xpath" select="'$p3, $p2, $p1'" />
  <xsl:value-of select="altova:evaluate($xpath)" />
  ```
  Sortie erreur : Aucune variable définie pour $p3.

encode-for-rtf [altova:]

altova:encode-for-rtf(input *as xs:string*, **preserveallwhitespace** *as xs:boolean*, **preservenewlines** *as xs:boolean*) **asxs:string** XSLT2 XSLT3

Convertit la chaîne d'entrée en tant que code pour RTF. Les espaces blancs et les nouvelles lignes seront préservés selon la valeur booléenne spécifiée pour leurs arguments respectifs.

[Haut 496]

Fonctions XBRL

Les fonctions XBRL Altova peuvent uniquement être utilisées avec des éditions des produits Altova qui présentent une prise en charge XBRL.

xbrl-footnotes [altova:]

altova:xbrl-footnotes(*node()***) asnode()*** XSLT2 XSLT3

Prend un nœud en tant que son argument d'entrée et retourne l'ensemble des nœuds de notes de pieds XBRL référencées par le nœud d'entrée.

xbrl-labels [altova:]

altova:xbrl-labels(*xs:QName*, *xs:string*) **asnode()*** XSLT2 XSLT3

Prend deux arguments d'entrée : un nom de nœud et l'emplacement de fichier de taxonomie contenant le nœud. La fonction retourne les nœuds de libellés XBRL associés avec le nœud d'entrée.

[Haut 496]

12.2.2.1.2 Fonctions XPath/XQuery : Date et heure

Les fonctions d'extension date/heure d'Altova peuvent être utilisées dans les expressions XPath et XQuery et fournissent des fonctions supplémentaires pour le traitement des données contenues en tant que les types de données de date et d'heures variés de XML Schema. Les fonctions dans cette section peuvent être utilisées avec les moteurs **XPath 3.0** et **XQuery 3.0** d'Altova. Ils sont disponibles dans des contextes XPath/XQuery.

Note concernant le nommage de fonctions et de l'applicabilité de la langue

Les fonctions d'extension Altova peuvent être utilisées dans les expressions XPath/XQuery. Elles fournissent des fonctions supplémentaires aux fonctions d'ores et déjà disponibles dans la librairie standard des fonctions XPath, XQuery et XSLT. Les fonctions d'extension Altova se trouvent dans **l'espace de nom des fonctions d'extension Altova, `http://www.altova.com/xslt-extensions`**, et sont indiquées dans cette section par le préfixe **`altova:`**, qui est présupposé être lié à cet espace de nom. Veuillez noter que, en ce qui concerne les versions futures de votre produit, la prise en charge d'une fonction peut être interrompue et le comportement de certaines fonctions peut changer. Veuillez consulter la documentation lors des publications à venir pour plus d'informations concernant la prise en charge des fonctions d'extension Altova de cette version.

Fonctions XPath (utilisées dans les expressions XPath dans XSLT) :	XP1 XP2 XP3.1
Fonctions XSLT (utilisées dans les expressions XPath dans XSLT) :	XSLT1 XSLT2 XSLT3
Fonctions XQuery (utilisées dans les expressions XQuery dans XQuery) :	XQ1 XQ3.1

▼ Regroupées selon les fonctionnalités

▼ Liste alphabétique

[Haut 499]

Ajouter une durée à xs:dateTime XP3.1 XQ3.1

Ces fonctions ajoutent une durée à **xs:dateTime** et retournent **xs:dateTime**. Le type xs:dateTime a un format de CCYY-MM-DDThh:mm:ss.sss. Il s'agit d'une concaténation des formats xs:date et xs:time séparés par la lettre T. Un suffixe de fuseau horaire (**+01:00**, par exemple) est optionnel.

add-years-to-dateTime [altova:]

altova:add-years-to-dateTime(DateTime *as xs:dateTime*, **Years** *as xs:integer***)**
asxs:dateTime XP3.1 XQ3.1

Ajoute une durée en années à xs:dateTime (*voir exemples ci-dessous*). Le deuxième argument est le nombre d'années à être ajouté à xs:dateTime fourni en tant que le premier argument. Le résultat est de type xs:dateTime.

Exemples

- **altova:add-years-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), 10) retourne 2024-01-15T14:00:00
- **altova:add-years-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), -4) retourne 2010-01-15T14:00:00

add-months-to-dateTime [altova:]

altova:add-months-to-dateTime(DateTime *as xs:dateTime*, **Months** *as xs:integer*)
asxs:dateTime XP3.1 XQ3.1
Ajoute une durée en mois à `xs:dateTime` (*voir exemples ci-dessous*). Le deuxième argument est le nombre de mois à être ajouté à `xs:dateTime` fourni en tant que le premier argument. Le résultat est de type `xs:dateTime`.

Exemples

- **altova:add-months-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), 10) retourne 2014-11-15T14:00:00
- **altova:add-months-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), -2) retourne 2013-11-15T14:00:00

add-days-to-dateTime [altova:]

altova:add-days-to-dateTime(DateTime *as xs:dateTime*, **Days** *as xs:integer*) **asxs:dateTime** XP3.1 XQ3.1
Ajoute une durée en jours à `xs:dateTime` (*voir exemples ci-dessous*). Le deuxième argument est le nombre de jours à être ajouté à `xs:dateTime` fourni en tant que le premier argument. Le résultat est de type `xs:dateTime`.

Exemples

- **altova:add-days-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), 10) retourne 2014-01-25T14:00:00
- **altova:add-days-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), -8) retourne 2014-01-07T14:00:00

add-hours-to-dateTime [altova:]

altova:add-hours-to-dateTime(DateTime *as xs:dateTime*, **Hours** *as xs:integer*)
asxs:dateTime XP3.1 XQ3.1
Ajoute une durée en heures à `xs:dateTime` (*voir exemples ci-dessous*). Le deuxième argument est le nombre d'heures à être ajouté à `xs:dateTime` fourni en tant que le premier argument. Le résultat est de type `xs:dateTime`.

Exemples

- **altova:add-hours-to-dateTime**(xs:dateTime("2014-01-15T13:00:00"), 10) retourne 2014-01-15T23:00:00
- **altova:add-hours-to-dateTime**(xs:dateTime("2014-01-15T13:00:00"), -8) retourne 2014-01-15T05:00:00

add-minutes-to-dateTime [altova:]

altova:add-minutes-to-dateTime(DateTime *as xs:dateTime*, **Minutes** *as xs:integer*)
asxs:dateTime XP3.1 XQ3.1
Ajoute une durée en minutes à `xs:dateTime` (*voir exemples ci-dessous*). Le deuxième argument est le nombre of minutes à être ajouté à `xs:dateTime` fourni en tant que le premier argument. Le résultat est de

type `xs:dateTime`.

Exemples

- **`altova:add-minutes-to-dateTime`**`(xs:dateTime("2014-01-15T14:10:00"), 45)` retourne `2014-01-15T14:55:00`
- **`altova:add-minutes-to-dateTime`**`(xs:dateTime("2014-01-15T14:10:00"), -5)` retourne `2014-01-15T14:05:00`

add-seconds-to-dateTime [altova:]

`altova:add-seconds-to-dateTime(DateTime` *`as xs:dateTime`*`, Seconds` *`as xs:integer`*`)` **`asxs:dateTime`** XP3.1 XQ3.1

Ajoute une durée en secondes à `xs:dateTime` (*voir exemples ci-dessous*). Le deuxième argument est le nombre de secondes à être ajouté à `xs:dateTime` fourni en tant que le premier argument. Le résultat est de type `xs:dateTime`.

Exemples

- **`altova:add-seconds-to-dateTime`**`(xs:dateTime("2014-01-15T14:00:10"), 20)` retourne `2014-01-15T14:00:30`
- **`altova:add-seconds-to-dateTime`**`(xs:dateTime("2014-01-15T14:00:10"), -5)` retourne `2014-01-15T14:00:05`

[Haut 499]

Ajouter une durée à xs: date XP3.1 XQ3.1

Ces fonctions ajoutent une durée à **`xs: date`** et retournent **`xs: date`**. Le type `xs:date` a un format `CCYY-MM-DD`.

add-years-to-date [altova:]

`altova:add-years-to-date(Date` *`as xs:date`*`, Years` *`as xs:integer`*`)` **`asxs:date`** XP3.1 XQ3.1

Ajoute une durée en années à une date. Le deuxième argument est le nombre d'années à être ajouté à `xs:date` fourni en tant que le premier argument. Le résultat est de type `xs:date`.

Exemples

- **`altova:add-years-to-date`**`(xs:date("2014-01-15"), 10)` retourne `2024-01-15`
- **`altova:add-years-to-date`**`(xs:date("2014-01-15"), -4)` retourne `2010-01-15`

add-months-to-date [altova:]

`altova:add-months-to-date(Date` *`as xs:date`*`, Months` *`as xs:integer`*`)` **`asxs:date`** XP3.1 XQ3.1

Ajoute une durée en mois à une date. Le deuxième argument est le nombre de mois à être ajouté à `xs:date` fourni en tant que le premier argument. Le résultat est de type `xs:date`.

Exemples

- **`altova:add-months-to-date`**`(xs:date("2014-01-15"), 10)` retourne `2014-11-15`
- **`altova:add-months-to-date`**`(xs:date("2014-01-15"), -2)` retourne `2013-11-15`

add-days-to-date [altova:]

altova:add-days-to-date(Date *as xs:date*, **Days** *as xs:integer*) **asxs:date** XP3.1 XQ3.1
Ajoute une durée en jours à une date. Le deuxième argument est le nombre de jours à être ajouté à `xs:date` fourni en tant que le premier argument. Le résultat est de type `xs:date`.

Exemples

- **altova:add-days-to-date**(xs:date("2014-01-15"), 10) retourne 2014-01-25
- **altova:add-days-to-date**(xs:date("2014-01-15"), -8) retourne 2014-01-07

[Haut 499]

Formater et récupérer des durées XP3.1 XQ3.1

Ces fonctions parsent une entrée **xs:duration** ou **xs:string** et retournent respectivement un **xs:string** ou **xs:duration**.

format-duration [altova:]

altova:format-duration(Duration *as xs:duration*, **Picture** *as xs:string*) **asxs:string** XP3.1 XQ3.1
Formate une durée qui est soumise en tant que le premier argument, selon une chaîne d'image soumise en tant que le second argument. La sortie est une chaîne de texte formatée conformément à la chaîne d'image.

Exemples

- **altova:format-duration**(xs:duration("P2DT2H53M11.7S"), "Days:[D01] Hours:[H01] Minutes:[m01] Seconds:[s01] Fractions:[f0]") retourne "Days:02 Hours:02 Minutes:53 Seconds:11 Fractions:7"
- **altova:format-duration**(xs:duration("P3M2DT2H53M11.7S"), "Months:[M01] Days:[D01] Hours:[H01] Minutes:[m01]") retourne "Months:03 Days:02 Hours:02 Minutes:53"

parse-duration [altova:]

altova:parse-duration(InputString *as xs:string*, **Picture** *as xs:string*) **asxs:duration** XP3.1 XQ3.1
Prend un patterned string en tant que le premier argument et une chaîne image en tant que le second argument. La chaîne d'entrée est parsée sur la base de la chaîne d'image et une `xs:duration` est retournée.

Exemples

- **altova:parse-duration**("Days:02 Hours:02 Minutes:53 Seconds:11 Fractions:7", "Days:[D01] Hours:[H01] Minutes:[m01] Seconds:[s01] Fractions:[f0]") retourne "P2DT2H53M11.7S"
- **altova:parse-duration**("Months:03 Days:02 Hours:02 Minutes:53 Seconds:11 Fractions:7", "Months:[M01] Days:[D01] Hours:[H01] Minutes:[m01]") retourne "P3M2DT2H53M"

[Haut 499]

Ajouter une durée à xs:time XP3.1 XQ3.1

Ces fonctions ajoutent une durée à **xs:time** et retournent **xs:time**. Le type `xs:time` a une forme lexicale de `hh:mm:ss.sss`. Un fuseau horaire en option peut être suffixé. La lettre `Z` indique le Temps universel coordonné (UTC). Tous les autres fuseaux horaires sont représentés par leur différence de l'UTC dans le format `+hh:mm`, ou `-hh:mm`. Si aucune valeur de fuseau horaire n'est présente, elle est considérée inconnue ; elle n'est pas considérée être UTC.

add-hours-to-time [altova:]

altova:add-hours-to-time(Time *as xs:time*, **Hours** *as xs:integer*) **asxs:time** XP3.1 XQ3.1

Ajoute une durée en heures à une heure de temps. Le deuxième argument est le nombre d'heures à être ajouté à `xs:time` fourni en tant que le premier argument. Le résultat est de type `xs:time`.

Exemples

- **altova:add-hours-to-time**(xs:time("11:00:00"), 10) retourne 21:00:00
- **altova:add-hours-to-time**(xs:time("11:00:00"), -7) retourne 04:00:00

add-minutes-to-time [altova:]

altova:add-minutes-to-time(Time *as xs:time*, **Minutes** *as xs:integer*) **asxs:time** XP3.1 XQ3.1

Ajoute une durée en minutes à une heure. Le deuxième argument est le nombre de minutes à être ajouté à `xs:time` fourni en tant que le premier argument. Le résultat est de type `xs:time`.

Exemples

- **altova:add-minutes-to-time**(xs:time("14:10:00"), 45) retourne 14:55:00
- **altova:add-minutes-to-time**(xs:time("14:10:00"), -5) retourne 14:05:00

add-seconds-to-time [altova:]

altova:add-seconds-to-time(Time *as xs:time*, **Minutes** *as xs:integer*) **asxs:time** XP3.1 XQ3.1

Ajoute une durée en secondes à une heure. Le deuxième argument est le nombre de secondes à être ajouté à `xs:time` fourni en tant que le premier argument. Le résultat est de type `xs:time`. Le composant Secondes peut être contenu dans une plage de `0` à `59.999`.

Exemples

- **altova:add-seconds-to-time**(xs:time("14:00:00"), 20) retourne 14:00:20
- **altova:add-seconds-to-time**(xs:time("14:00:00"), 20.895) retourne 14:00:20.895

[Haut 499]

Supprimer la partie du fuseau horaire des types de données date/heures XP3.1 XQ3.1

Ces fonctions permettent de supprimer le fuseau horaire des valeurs **xs:dateTime**, **xs:date** ou **xs:time** actuelles, respectivement. Veuillez noter que la différence entre `xs:dateTime` et `xs:dateTimeStamp` est que dans le cas de ce dernier, la partie fuseau horaire est requise (alors qu'elle est optionnelle dans le premier des deux cas). Donc, le format d'une valeur `xs:dateTimeStamp` est : `CCYY-MM-DDThh:mm:ss.sss±hh:mm`. ou `CCYY-MM-DDThh:mm:ss.sssZ`. Si la date et l'heure sont lues depuis l'horloge du système, en tant que

`xs:dateTimeStamp`, la fonction `current-dateTime-no-TZ()` peut être utilisée pour supprimer le fuseau horaire s'il est requis.

current-date-no-TZ [altova:]

`altova:current-date-no-TZ()` **`asxs:date`** **XP3.1** **XQ3.1**
Cette fonction ne prend aucun argument. Elle supprime la partie fuseau horaire de `current-date()` (qui est la date actuelle selon l'horloge système) et retourne une valeur `xs:date`.

Exemples

Si la date actuelle est **`2014-01-15+01:00`**:

- **`altova:current-date-no-TZ`**`()` retourne `2014-01-15`

current-dateTime-no-TZ [altova:]

`altova:current-dateTime-no-TZ()` **`asxs:dateTime`** **XP3.1** **XQ3.1**
Cette fonction ne prend aucun argument. Elle supprime la partie fuseau horaire de `current-dateTime()` (qui est la date-heure actuelle selon l'horloge système) et retourne une valeur`xs:dateTime`.

Exemples

Si la date-heure actuelle est **`2014-01-15T14:00:00+01:00`**:

- **`altova:current-dateTime-no-TZ`**`()` retourne `2014-01-15T14:00:00`

current-time-no-TZ [altova:]

`altova:current-time-no-TZ()` **`asxs:time`** **XP3.1** **XQ3.1**
Cette fonction ne prend aucun argument. Elle supprime la partie fuseau horaire de `current-time()` (qui est l'heure actuelle selon l'horloge système) et retourne une valeur `xs:time`.

Exemples

Si l'heure actuelle est **`14:00:00+01:00`**:

- **`altova:current-time-no-TZ`**`()` retourne `14:00:00`

date-no-TZ [altova:]

`altova:date-no-TZ(InputDate` *`as xs:date`***`)`** **`asxs:date`** **XP3.1** **XQ3.1**
Cette fonction prend un argument `xs:date`, en supprime la partie fuseau horaire et retourne une valeur `xs:date`. Veuillez noter que la date n'est pas modifiée.

Exemples

- **`altova:date-no-TZ`**`(xs:date("2014-01-15+01:00"))` retourne `2014-01-15`

dateTime-no-TZ [altova:]

`altova:dateTime-no-TZ(InputDateTime` *`as xs:dateTime`***`)`** **`asxs:dateTime`** **XP3.1** **XQ3.1**
Cette fonction prend un argument `xs:dateTime`, en supprime la partie fuseau horaire, et retourne une valeur `xs:dateTime`. Veuillez noter que ni la date ni l'heure n'est modifiée.

Exemples

- **`altova:dateTime-no-TZ`**`(xs:date("2014-01-15T14:00:00+01:00"))` retourne `2014-01-15T14:00:00`

time-no-TZ [altova:]

altova:time-no-TZ(InputTime *as xs:time*) **asxs:time** XP3.1 XQ3.1
Cette fonction prend un argument `xs:time`, en supprime la partie de fuseau horaire, et retourne une valeur `xs:time`. Veuillez noter que l'heure n'est pas modifiée.

Exemples

- **altova:time-no-TZ**(xs:time("14:00:00+01:00")) retourne 14:00:00

[Haut 499]

Retourne le nombre de jours, d'heures, de minutes, de secondes des durées XP3.1 XQ3.1

Ces fonctions retournent le nombre de jours dans un mois, et le nombre d'heures, de minutes et de secondes, respectivement depuis les durées.

days-in-month [altova:]

altova:days-in-month(Year *as xs:integer*, **Month** *as xs:integer*) **asxs:integer** XP3.1 XQ3.1
Retourne le nombre de jours dans le mois spécifié. Le mois est spécifié avec les arguments `Year` et `Month`.

Exemples

- **altova:days-in-month**(2018, 10) retourne 31
- **altova:days-in-month**(2018, 2) retourne 28
- **altova:days-in-month**(2020, 2) retourne 29

hours-from-dayTimeDuration-accumulated

altova:hours-from-dayTimeDuration-accumulated(DayAndTime *as xs:duration*) **asxs:integer** XP3.1 XQ3.1
Retourne le nombre total d'heures dans la durée soumise par l'argument `DayAndTime` (qui est de type `xs:duration`). Les heures dans les composants `Day` et `Time` sont additionnés pour donner un résultat qui est un entier. Le décompte d'une nouvelle heure dure uniquement 60 minutes complètes. Des durées négatives entraînent une valeur d'heures négative.

Exemples

- **altova:hours-from-dayTimeDuration-accumulated**(xs:duration("P5D")) retourne 120, qui est le nombre total d'heures dans 5 jours.
- **altova:hours-from-dayTimeDuration-accumulated**(xs:duration("P5DT2H")) retourne 122, qui est le nombre total d'heures dans 5 jours plus 2 heures.
- **altova:hours-from-dayTimeDuration-accumulated**(xs:duration("P5DT2H60M")) retourne 123, qui est le nombre total d'heures dans 5 jours plus 2 heures et 60 mins.
- **altova:hours-from-dayTimeDuration-accumulated**(xs:duration("P5DT2H119M")) retourne 123, qui est le nombre total d'heures dans 5 jours plus 2 heures et 119 mins.
- **altova:hours-from-dayTimeDuration-accumulated**(xs:duration("P5DT2H120M")) retourne 124,qui est le nombre total d'heures dans 5 jours plus 2 heures et 120 mins.
- **altova:hours-from-dayTimeDuration-accumulated**(xs:duration("-P5DT2H")) retourne -122

minutes-from-dayTimeDuration-accumulated

altova:minutes-from-dayTimeDuration-accumulated(DayAndTime *as xs:duration*) **asxs:integer**
XP3.1 XQ3.1

Retourne le nombre total de minutes dans la durée soumise par l'argument `DayAndTime` (qui est de type `xs:duration`). Les minutes dans les composants `Day` et `Time` sont additionnés pour donner un résultat qui est un entier. Des durées négatives entraînent une valeur de minute négative.

Exemples

- **altova:minutes-from-dayTimeDuration-accumulated**(xs:duration("PT60M")) retourne 60
- **altova:minutes-from-dayTimeDuration-accumulated**(xs:duration("PT1H")) retourne 60, qui est le nombre total de minutes dans une heure.
- **altova:minutes-from-dayTimeDuration-accumulated**(xs:duration("PT1H40M")) retourne 100
- **altova:minutes-from-dayTimeDuration-accumulated**(xs:duration("P1D")) retourne 1440, qui est le nombre total de minutes dans un jour.
- **altova:minutes-from-dayTimeDuration-accumulated**(xs:duration("-P1DT60M")) retourne -1500

seconds-from-dayTimeDuration-accumulated

altova:seconds-from-dayTimeDuration-accumulated(DayAndTime *as xs:duration*) **asxs:integer**
XP3.1 XQ3.1

Retourne le nombre total de minutes dans la durée soumise par l'argument `DayAndTime` (qui est de type `xs:duration`). Les minutes dans les composants `Day` et `Time` sont additionnés pour donner un résultat qui est un entier. Des durées négatives entraînent une valeur de minute négative .

Exemples

- **altova:seconds-from-dayTimeDuration-accumulated**(xs:duration("PT1M")) retourne 60, qui est le nombre total de secondes dans une minute.
- **altova:seconds-from-dayTimeDuration-accumulated**(xs:duration("PT1H")) retourne 3600, qui est le nombre total de secondes dans une heure.
- **altova:seconds-from-dayTimeDuration-accumulated**(xs:duration("PT1H2M")) retourne 3720
- **altova:seconds-from-dayTimeDuration-accumulated**(xs:duration("P1D")) retourne 86400, qui est le nombre total de secondes dans un jour.
- **altova:seconds-from-dayTimeDuration-accumulated**(xs:duration("-P1DT1M")) retourne -86460

Retourne le jour de la semaine à partir de xs:dateTime ou xs:date XP3.1 XQ3.1

Ces fonctions retournent le jour de la semaine (en tant qu'entier) depuis `xs:dateTime` ou `xs:date`. Les jours de la semaine sont numérotés (format américain) de `1` à `7`, avec `Sunday=1`. Dans le format européen, la semaine commence par Lundi (`=1`). Dans le format américain elle commence par `Sunday=1`. Configurer en utilisant l'entier `0` et où un entier est accepté pour indiquer le format.

weekday-from-dateTime [altova:]

altova:weekday-from-dateTime(DateTime *as xs:dateTime*) **asxs:integer** XP3.1 XQ3.1
Prend une date-avec-heure en tant que son seul argument et retourne le jour de la semaine de cette date sous forme d'un entier. Les jours de la semaine sont numérotés en commençant avec `Sunday=1`. Si le format européen est requis (où `Monday=1`), utiliser l'autre signature de cette fonction (*voir signature suivante ci-dessous*).

Exemples

- **altova:weekday-from-dateTime**(xs:dateTime("2014-02-03T09:00:00")) retourne 2, ce qui indique un lundi.

altova:weekday-from-dateTime(DateTime *as xs:dateTime*, **Format** *as xs:integer*) **asxs:integer** XP3.1 XQ3.1
Prend une date-avec-heure en tant que son premier argument et retourne le jour de la semaine de cette date sous forme d'un entier. Si le second argument (entier) est `0`, les jours de la semaine sont numérotés de `1` à `7` en commençant avec `Sunday=1`. Si le second argument est un entier différent de `0`, alors `Monday=1`. S'il n'y a pas de second argument, la fonction est lue comme possédant l'autre signature de cette fonction (*voir signature précédente*).

Exemples

- **altova:weekday-from-dateTime**(xs:dateTime("2014-02-03T09:00:00"), 1) retourne 1, ce qui indique un lundi
- **altova:weekday-from-dateTime**(xs:dateTime("2014-02-03T09:00:00"), 4) retourne 1, ce qui indique un lundi
- **altova:weekday-from-dateTime**(xs:dateTime("2014-02-03T09:00:00"), 0) retourne 2, ce qui indique un lundi.

weekday-from-date [altova:]

altova:weekday-from-date(Date *as xs:date*) **asxs:integer** XP3.1 XQ3.1
Prend une date en tant que son seul argument et retourne le jour de la semaine de cette date sous forme d'un entier. Les jours de la semaine sont numérotés en commençant avec `Sunday=1`. Si le format européen est requis (où `Monday=1`), utiliser l'autre signature de cette fonction (*voir signature suivante ci-dessous*).

Exemples

- **altova:weekday-from-date**(xs:date("2014-02-03+01:00")) retourne 2, ce qui indique un lundi.

altova:weekday-from-date(Date *as xs:date*, **Format** *as xs:integer*) **asxs:integer** XP3.1 XQ3.1
Prend une date en tant que son premier argument et retourne le jour de la semaine de cette date sous forme d'un entier. Si le second argument (`Format`) est `0`, les jours de la semaine sont numérotés de `1` à `7` en commençant avec `Sunday=1`. Si le second argument est un entier différent de `0`, alors `Monday=1`. S'il n'y a pas de second argument, la fonction est lue comme possédant l'autre signature de cette fonction (*voir signature précédente*).

Exemples

- **altova:weekday-from-date**(xs:date("2014-02-03"), 1) retourne 1, ce qui indique un lundi
- **altova:weekday-from-date**(xs:date("2014-02-03"), 4) retourne 1, ce qui indique un lundi
- **altova:weekday-from-date**(xs:date("2014-02-03"), 0) retourne 2, ce qui indique un lundi.

[Haut 499]

Retourne le nombre de la semaine à partir de xs:dateTime ou xs:date XP2 XQ1 XP3.1 XQ3.1

Ces fonctions retournent le nombre de la semaine (en tant qu'entier) depuis `xs:dateTime` ou `xs:date`. La numérotation des semaines est disponible dans les formats de calendrier US, ISO/European et Islamiques. La numérotation des semaines est différente dans ces formats de calendrier parce que la semaine est considérée démarrer avec un jour différent selon le format (dimanche pour le format US, lundi pour le format ISO/Européen, et samedi dans le format islamique).

weeknumber-from-date [altova:]

altova:weeknumber-from-date(Date *as xs:date*, **Calendar** *as xs:integer*) **asxs:integer** XP2 XQ1 XP3.1 XQ3.1

Retourne le numéro de la semaine de l'argument **Date** soumis en tant qu'entier. Le deuxième argument (**Calendar**) spécifie le système de calendrier à suivre.
Les valeurs de **Calendar** prises en charge sont :

- **0 = US calendar** (*semaine commence dimanche*)
- **1 = ISO standard, European calendar** (*semaine commence lundi*)
- **2 = Islamic calendar** (*semaine commence samedi*)

Le réglage par défaut est **0**.

Exemples

- **altova:weeknumber-from-date**(xs:date("2014-03-23"), 0) retourne 13
- **altova:weeknumber-from-date**(xs:date("2014-03-23"), 1) retourne 12
- **altova:weeknumber-from-date**(xs:date("2014-03-23"), 2) retourne 13
- **altova:weeknumber-from-date**(xs:date("2014-03-23")) retourne 13

Le jour de la date dans les exemples ci-dessus (2014-03-23) est dimanche. Les calendriers US et musulmans sont donc une semaine en avant par rapport au calendrier européen à ce jour.

weeknumber-from-dateTime [altova:]

altova:weeknumber-from-dateTime(DateTime *as xs:dateTime*, **Calendar** *as xs:integer*) **asxs:integer** XP2 XQ1 XP3.1 XQ3.1

Retourne le numéro de la semaine de l'argument **DateTime** soumis en tant qu'entier. Le deuxième argument (**Calendar**) spécifie le système de calendrier à suivre.
Les valeurs de **Calendar** prises en charge sont :

- **0 = US calendar** (*semaine commence dimanche*)
- **1 = ISO standard, European calendar** (*semaine commence lundi*)
- **2 = Islamic calendar** (*semaine commence samedi*)

Le réglage par défaut est **0**.

Exemples

- **altova:weeknumber-from-dateTime**(xs:dateTime("2014-03-23T00:00:00"), 0) retourne 13

- **altova:weeknumber-from-dateTime**(xs:dateTime("2014-03-23T00:00:00"), 1) retourne 12
- **altova:weeknumber-from-dateTime**(xs:dateTime("2014-03-23T00:00:00"), 2) retourne 13
- **altova:weeknumber-from-dateTime**(xs:dateTime("2014-03-23T00:00:00")) retourne 13

Le jour du dateTime dans les exemples ci-dessus (2014-03-23T00:00:00) est dimanche. Les calendriers US et musulmans sont donc une semaine en avant par rapport au calendrier européen à ce jour.

[Haut 499]

Construire le type de date, d'heure ou de durée à partir de leurs composants lexicaux XP3.1 XQ3.1

Les fonctions prennent les composants lexicaux du type de données xs:date, xs:time ou xs:duration en tant qu'arguments d'entrée et les combinent pour construire le type de données respectif.

build-date [altova:]

altova:build-date(Year *as xs:integer*, **Month** *as xs:integer*, **Date** *as xs:integer***)**
asxs:date XP3.1 XQ3.1

Les premier, second et troisième arguments sont respectivement l'année, le mois et la date. Ils sont combinés pour construire une valeur de type xs:date. Les valeurs de l'entier doivent se situer dans le cadre de la plage correcte de cette partie de la date. Par exemple, le deuxième argument (pour la partie du mois) ne devrait pas être supérieur à 12.

Exemples

- **altova:build-date**(2014, 2, 03) retourne 2014-02-03

build-time [altova:]

altova:build-time(Hours *as xs:integer*, **Minutes** *as xs:integer*, **Seconds** *as xs:integer***)**
asxs:time XP3.1 XQ3.1

Les premiers, seconds et troisièmes arguments sont, respectivement, les valeurs d'heure (0 to 23), de minutes (0 to 59) et de secondes (0 to 59). Ils sont combinés pour construire une valeur de type xs:time. Les valeurs des entiers doivent se trouver dans le cadre de la plage correcte de cette partir de temps en particulier. Par exemple, le deuxième argument (Minutes) ne devrait pas être supérieur à 59. Pour ajouter une partie fuseau horaire à la valeur, utiliser l'autre signature de cette fonction (*voir signature suivante*).

Exemples

- **altova:build-time**(23, 4, 57) retourne 23:04:57

altova:build-time(Hours *as xs:integer*, **Minutes** *as xs:integer*, **Seconds** *as xs:integer*, **TimeZone** *as xs:string***)** **asxs:time** XP3.1 XQ3.1

Les premiers, seconds et troisièmes arguments sont, respectivement, les valeurs d'heure (0 to 23), de minutes (0 to 59) et de secondes (0 to 59). Le quatrième argument est une chaîne qui fournit la partie fuseau horaire de la valeur. Les quatre arguments sont combinés pour construire une valeur de type xs:time. Les valeurs des entiers doivent se trouver dans le cadre de la plage correcte de cette partie de temps en particulier. Par exemple, le deuxième argument (Minutes) ne doit pas être supérieur à 59.

Exemples

- **altova:build-time**(23, 4, 57, '+1') retourne 23:04:57+01:00

build-duration [altova:]

altova:build-duration(Years *as xs:integer*, **Months** *as xs:integer*) **asxs:yearMonthDuration** XP3.1 XQ3.1

Prend deux arguments pour construire une valeur de type `xs:yearMonthDuration`. Les premiers arguments fournissent la partie `Years` de la valeur de durée, alors que le deuxième argument fournie la partie `Months`. Si le deuxième argument (`Months`) est supérieur ou égale à `12`, alors l'entier est divisé par `12`; le quotient est ajouté au premier argument pour fournir la partie `Years` de la valeur de durée alors que le reste (de la division) fournit la partie `Months`. Pour construire une durée de type `xs:dayTimeDuration`., voir la signature suivante.

Exemples

- **altova:build-duration**(2, 10) retourne P2Y10M
- **altova:build-duration**(14, 27) retourne P16Y3M
- **altova:build-duration**(2, 24) retourne P4Y

altova:build-duration(Days *as xs:integer*, **Hours** *as xs:integer*, **Minutes** *as xs:integer*, **Seconds** *as xs:integer*) as **xs:dayTimeDuration** XP3.1 XQ3.1

Prend quatre arguments et les combine pour construire une valeur de type `xs:dayTimeDuration`. Le premier argument fournit la partie `Days` de la valeur de durée, le deuxième, troisième et quatrième argument fournit respectivement les parties `Hours`, `Minutes` et `Seconds` de la valeur de durée. Chacun des trois arguments Time est converti en une valeur équivalente en termes de l'unité suivante plus élevée et le résultat est utilisé pour le calcul d'une valeur de durée totale. Par exemple, 72 secondes est converti en `1M+12S` (1 minute et 12 secondes), et cette valeur est utilisée pour le calcul de la valeur de durée totale. Pour construire une durée de type `xs:yearMonthDuration`., voir la signature précédente.

Exemples

- **altova:build-duration**(2, 10, 3, 56) retourne P2DT10H3M56S
- **altova:build-duration**(1, 0, 100, 0) retourne P1DT1H40M
- **altova:build-duration**(1, 0, 0, 3600) retourne P1DT1H

[Haut 499]

Construire le type de date, dateHeure ou heure à partir de l'entrée de chaîne XP2 XQ1 XP3.1 XQ3.1

Ces fonctions prennent des chaînes en tant qu'arguments et construisent des types de données `xs:date`, `xs:dateTime`, ou `xs:time`. La chaîne est analysée pour les composants du type de données basé sur un argument de modèle soumis.

parse-date [altova:]

altova:parse-date(Date *as xs:string*, **DatePattern** *as xs:string*) **asxs:date** XP2 XQ1 XP3.1 XQ3.1

Retourne la chaîne d'entrée **Date** en tant qu'une valeur **xs:date**. Le deuxième argument **DatePattern** spécifie le modèle (séquence des composants) de la chaîne d'entrée. **DatePattern** est décrit avec les spécificateurs de composants regroupés ci-dessous et avec les séparateurs de composant qui peuvent être n'importe quel caractère. Voir les exemples ci-dessous.

D	Jour
M	Mois

Y	Année

Le modèle dans **DatePattern** doit correspondre au modèle dans **Date**. Puisque la sortie est de type **xs:date**, la sortie aura toujours le format lexical **YYYY-MM-DD**.

Exemples

- **altova:parse-date**(xs:string("09-12-2014"), "[D]-[M]-[Y]") retourne 2014-12-09
- **altova:parse-date**(xs:string("09-12-2014"), "[M]-[D]-[Y]") retourne 2014-09-12
- **altova:parse-date**("06/03/2014", "[M]/[D]/[Y]") retourne 2014-06-03
- **altova:parse-date**("06 03 2014", "[M] [D] [Y]") retourne 2014-06-03
- **altova:parse-date**("6 3 2014", "[M] [D] [Y]") retourne 2014-06-03

parse-dateTime [altova:]

altova:parse-dateTime(DateTime *as xs:string*, **DateTimePattern** *as xs:string***)**
asxs:dateTime XP2 XQ1 XP3.1 XQ3.1
Retourne la chaîne d'entrée **DateTime** en tant que valeur **xs:dateTime**. Le deuxième argument **DateTimePattern** spécifie le modèle (séquence des composants) de la chaîne d'entrée. **DateTimePattern** est décrit avec les spécificateurs de composants regroupés ci-dessous et avec les séparateurs de composant qui peuvent être n'importe quel caractère. Voir les exemples ci-dessous.

D	Date
M	Mois
Y	Année
H	Heure
m	Minutes
s	Secondes

Le modèle dans **DateTimePattern** doit correspondre au modèle dans **DateTime**. Puisque la sortie est de type **xs:dateTime**, la sortie aura toujours le format lexical **YYYY-MM-DDTHH:mm:ss**.

Exemples

- **altova:parse-dateTime**(xs:string("09-12-2014 13:56:24"), "[M]-[D]-[Y] [H]:[m]:[s]") retourne 2014-09-12T13:56:24
- **altova:parse-dateTime**("time=13:56:24; date=09-12-2014", "time=[H]:[m]:[s]; date=[D]-[M]-[Y]") retourne 2014-12-09T13:56:24

parse-time [altova:]

altova:parse-time(Time *as xs:string*, **TimePattern** *as xs:string***) asxs:time** XP2 XQ1 XP3.1 XQ3.1
Retourne la chaîne d'entrée **Time** en tant qu'une valeur **xs:time**.Le deuxième argument **TimePattern** spécifie le modèle (séquence des composants) de la chaîne d'entrée. **TimePattern** est décrit avec les spécificateurs de composants regroupés ci-dessous et avec les séparateurs de composant qui peuvent être n'importe quel caractère. Voir les exemples ci-dessous.

H	Heure
m	minutes

s secondes

Le modèle dans **TimePattern** doit correspondre au modèle dans **Time**. Puisque la sortie est de type **xs:time**, la sortie aura toujours le format lexical **HH:mm:ss**.

Exemples

- **altova:parse-time**(xs:string("13:56:24"), "[H]:[m]:[s]") retourne 13:56:24
- **altova:parse-time**("13-56-24", "[H]-[m]") retourne 13:56:00
- **altova:parse-time**("time=13h56m24s", "time=[H]h[m]m[s]s") retourne 13:56:24
- **altova:parse-time**("time=24s56m13h", "time=[s]s[m]m[H]h") retourne 13:56:24

[Haut 499]

Fonctions liées à l'âge XP3.1 XQ3.1

Ces fonctions retournent l'âge tel que calculé (i) entre une date d'argument d'entrée et la date actuelle, ou (ii) entre deux dates d'argument d'entrée. La fonction **altova:age** retourne l'âge en termes d'années, la fonction **altova:age-details** retourne l'âge en tant qu'une séquence de trois entiers indiquant les années, mois et jours de l'âge.

age [altova:]

altova:age(StartDate *as xs:date*) **asxs:integer** XP3.1 XQ3.1
Retourne un entier représentant l'âge *en années* d'un objet, en comptant depuis une date de départ soumise en tant que l'argument et se terminant avec la date actuelle (prise depuis l'horloge système). Si l'argument d'entrée est une date supérieure ou égale à une année dans le futur, la valeur de retour sera négative.

Exemples

Si la date actuelle est **2014-01-15** :

- **altova:age**(xs:date("2013-01-15")) retourne 1
- **altova:age**(xs:date("2013-01-16")) retourne 0
- **altova:age**(xs:date("2015-01-15")) retourne -1
- **altova:age**(xs:date("2015-01-14")) retourne 0

altova:age(StartDate *as xs:date*, **EndDate** *as xs:date*) **asxs:integer** XP3.1 XQ3.1
Retourne un entier représentant l'âge *en années* d'un objet, en comptant depuis une date de départ soumise en tant que l'argument jusqu'à une date de fin qui est de deuxième argument. La valeur de retour sera négative si le premier argument est tardif d'une année ou plus que le deuxième argument.

Exemples

Si la date actuelle est **2014-01-15**:

- **altova:age**(xs:date("2000-01-15"), xs:date("2010-01-15")) retourne 10
- **altova:age**(xs:date("2000-01-15"), current-date()) retourne 14 si la date actuelle est 2014-01-15
- **altova:age**(xs:date("2014-01-15"), xs:date("2010-01-15")) retourne -4

age-details [altova:]

altova:age-details(InputDate *as xs:date*) as (**xs:integer)*** XP3.1 XQ3.1
Retourne trois entiers qui sont respectivement les années, les mois et les jours entre la date soumise en tant que l'argument et la date actuelle (prise depuis l'horloge système). Le résultat de la somme de `years+months+days` donne le total de la différence de temps entre les deux dates (la date d'entrée et la date actuelle). La date d'entrée peut avoir une valeur précédant ou succédant à la date actuelle mais que la date d'entrée soit précédente ou succédant n'est pas indiqué par le signe des valeurs de retour ; les valeurs de retour sont toujours positives.

Exemples

Si la date actuelle est **2014-01-15**:

- `altova:age-details(xs:date("2014-01-16"))` retourne `(0 0 1)`
- `altova:age-details(xs:date("2014-01-14"))` retourne `(0 0 1)`
- `altova:age-details(xs:date("2013-01-16"))` retourne `(1 0 1)`
- `altova:age-details(current-date())` retourne `(0 0 0)`

altova:age-details(Date-1 *as xs:date*, **Date-2** *as xs:date*) as (**xs:integer)*** XP3.1 XQ3.1
Retourne trois entiers qui sont respectivement les années, les mois et les jours entre les deux dates d'argument. Le résultat de la somme de `years+months+days` donne le total de la différence de temps entre les deux dates d'entrée ; peu importe que la date soit la précédente ou la subséquente des deux dates, elle est soumise en tant que le premier argument. Les valeurs de retour n'indiquent pas si la date d'entrée se produit avant ou après la date actuelle. Les valeurs de retour sont toujours positives.

Exemples

- `altova:age-details(xs:date("2014-01-16"), xs:date("2014-01-15"))` retourne `(0 0 1)`
- `altova:age-details(xs:date("2014-01-15"), xs:date("2014-01-16"))` retourne `(0 0 1)`

[Haut 499]

Fonctions Epoch time (heure Unix) XP3.1 XQ3.1

Epoch time est un système horaire utilisé dans les systèmes Unix. Il définit tout moment donné comme étant le nombre de secondes écoulées depuis 00:00:00 UTC le 1er janvier 1970. Ces fonctions Epoch time convertissent les valeurs **xs:dateTime** en valeurs Epoch time et vice versa.

dateTime-from-epoch [altova:]

altova:dateTime-from-epoch(Epoch *as xs:decimal* **as xs:dateTime** XP3.1 XQ3.1
Epoch time est un système horaire utilisé sur les systèmes Unix. Il définit tout moment donné comme étant l nombre de secondes écoulées depuis 00:00:00 UTC le 1er janvier 1970. La fonction **dateTime-from-epoch** retourne l'équivalent **xs:dateTime** d'un Epoch time, l'ajuste pour son fuseau horaire local et inclut l'information du fuseau horaire dans le résultat.

La fonction prend un argument `xs:decimal` et retourne une valeur `xs:dateTime` qui inclut une partie (fuseau horaire) **TZ**. Le résultat est obtenu en calculant l'équivalent UTC **dateTime** de Epoch time, et en l'ajoutant à son fuseau horaire local (pris de l'horloge système). Par exemple, si la fonction est exécutée sur un appareil qui a été défini pour être dans un fuseau horaire +01:00 (relatif à UTC), après avoir calculé l'équivalent UTC **dateTime**, une heure sera ajoutée au résultat. L'information du fuseau horaire, qui est une partie lexicale optionnelle du résultat **xs:dateTime**, est également rapportée dans le résultat **dateTime**. Comparez ce résultat avec celui de **dateTime-from-epoch-no-TZ**, et consultez également la fonction **epoch-from-dateTime**.

Exemples

Les exemples ci-dessous supposent un fuseau horaire local UTC +01:00. En conséquence, l'équivalent UTC **dateTime** de l'Epoch time soumis sera incrémenté d'une heure. Le fuseau horaire est rapporté dans le résultat.

- **altova:dateTime-from-epoch**(34) retourne 1970-01-01T01:00:34+01:00
- **altova:dateTime-from-epoch**(62) retourne 1970-01-01T01:01:02+01:00

dateTime-from-epoch-no-TZ [altova:]

altova:dateTime-from-epoch-no-TZ(Epoch *as xs:decimal* **as xs:dateTime** XP3.1 XQ3.1
Epoch time est un système horaire utilisé sur les systèmes Unix. Il définit tout moment donné comme étant l nombre de secondes écoulées depuis 00:00:00 UTC le 1er janvier 1970. La fonction **dateTime-from-epoch-no-TZ** retourne l'équivalent **xs:dateTime** d'un Epoch time, l'ajuste pour son fuseau horaire local, mais n'inclut pas l'information du fuseau horaire dans le résultat.

La fonction prend un xs:decimal argument et retourne une valeur xs:dateTime qui n'inclut pas de partie (fuseau horaire) **TZ**. Le résultat est obtenu en calculant l'équivalent UTC **dateTime** de Epoch time, et en l'ajoutant à son fuseau horaire local (pris de l'horloge système). Par exemple, si la fonction est exécutée sur un appareil qui a été défini pour être dans un fuseau horaire +01:00 (relatif à UTC), après avoir calculé l'équivalent, une heure sera ajoutée au résultat. L'information du fuseau horaire, qui est une partie lexicale optionnelle du résultat **xs:dateTime**, n'est pas rapportée dans le résultat **dateTime**. Comparez ce résultat avec celui de **dateTime-from-epoch**, et consultez également la fonction **epoch-from-dateTime**.

Exemples

Les exemples ci-dessous supposent un fuseau horaire local UTC +01:00. En conséquence, l'équivalent UTC **dateTime** de l'Epoch time soumis sera incrémenté d'une heure. Le fuseau horaire n'est pas rapporté dans le résultat.

- **altova:dateTime-from-epoch**(34) returns 1970-01-01T01:00:34
- **altova:dateTime-from-epoch**(62) returns 1970-01-01T01:01:02

epoch-from-dateTime [altova:]

altova:epoch-from-dateTime(dateTimeValue *as xs:dateTime***) as xs:decimal** XP3.1 XQ3.1
Epoch time est un système horaire utilisé sur les systèmes Unix. Il définit tout moment donné comme étant l nombre de secondes écoulées depuis 00:00:00 UTC le 1er janvier 1970. La fonction **epoch-from-dateTime** retourne l'équivalent **xs:dateTime** d'un Epoch time, l'ajuste pour son fuseau horaire local et inclut l'information du fuseau horaire dans le résultat. Veuillez noter que vous devrez explicitement construire la valeur **xs:dateTime**. La valeur soumise **xs:dateTime** peut ou ne peut pas contenir la partie optionnelle **TZ** (fuseau horaire).

Que la partie du fuseau horaire soit soumise en tant que partie de l'argument ou non, le décalage du fuseau horaire local (pris de l'horloge système) est retiré de l'argument soumis **dateTimeValue**. Ceci porduit l'heure UTC équivalente de laquelle l'Epoch time équivalent est calculé. Par exemple, si la fonction est exécutée sur un appareil qui a été défini pour être dans un fuseau horaire +01:00 (relatif à UTC), une heure sera retirée de la valeur soumise **dateTimeValue** avant de calculer la valeur Epoch. Consultez également la fonction **dateTime-from-epoch**.

Exemples

Les exemples ci-dessous supposent un fuseau horaire local UTC +01:00. En conséquence, une heure sera soustraite à **dateTime** avant de calculer l'Epoch time.

- **altova:epoch-from-dateTime**(xs:dateTime("1970-01-01T01:00:34+01:00")) returns 34
- **altova:epoch-from-dateTime**(xs:dateTime("1970-01-01T01:00:34")) returns 34
- **altova:epoch-from-dateTime**(xs:dateTime("2021-04-01T11:22:33")) returns 1617272553

[Haut 499]

12.2.2.1.3 Fonctions XPath/XQuery : Géolocalisation

Les fonctions d'extension de géolocalisation XPath/XQuery suivantes sont prises en charge dans la version actuelle de MapForce et peuvent être utilisées dans (i) des expressions XPath dans un contexte XSLT, ou (ii) des expressions XQuery dans un document XQuery.

Note concernant le nommage de fonctions et de l'applicabilité de la langue

Les fonctions d'extension Altova peuvent être utilisées dans les expressions XPath/XQuery. Elles fournissent des fonctions supplémentaires aux fonctions d'ores et déjà disponibles dans la librairie standard des fonctions XPath, XQuery et XSLT. Les fonctions d'extension Altova se trouvent dans **l'espace de nom des fonctions d'extension Altova, http://www.altova.com/xslt-extensions**, et sont indiquées dans cette section par le préfixe **altova:**, qui est présupposé être lié à cet espace de nom. Veuillez noter que, en ce qui concerne les versions futures de votre produit, la prise en charge d'une fonction peut être interrompue et le comportement de certaines fonctions peut changer. Veuillez consulter la documentation lors des publications à venir pour plus d'informations concernant la prise en charge des fonctions d'extension Altova de cette version.

Fonctions XPath (utilisées dans les expressions XPath dans XSLT) :	XP1 XP2 XP3.1
Fonctions XSLT (utilisées dans les expressions XPath dans XSLT) :	XSLT1 XSLT2 XSLT3
Fonctions XQuery (utilisées dans les expressions XQuery dans XQuery) :	XQ1 XQ3.1

format-geolocation [altova:]

altova:format-geolocation(Latitude *as xs:decimal*, **Longitude** *as xs:decimal*, **GeolocationOutputStringFormat** *as xs:integer*) **as**xs:string

XP3.1 XQ3.1

Prend la latitude et la longitude en tant que les deux premiers arguments, et sort la géolocalisation en tant que chaîne. Le troisième argument, **GeolocationOutputStringFormat**, est le format de la chaîne de sortie de géolocalisation ; il utilise des valeurs d'entier allant de 1 à 4 pour identifier le format de chaîne de sortie (*voir 'Formats de chaîne de sortie de géolocalisation' ci-dessous*). Les valeurs de latitude vont de +90 à -90 (N à S). Les valeurs de longitude vont de +180 à -180 (E à O).

Note : La fonction `image-exif-data` 525 et les attributs de métadonnées Exif peuvent être utilisés pour fournie les chaînes d'entrée.

- *Exemples*
  - **`altova:format-geolocation`**`(33.33, -22.22, 4)` retourne `xs:string "33.33 -22.22"`
  - **`altova:format-geolocation`**`(33.33, -22.22, 2)` retourne `xs:string "33.33N 22.22W"`
  - **`altova:format-geolocation`**`(-33.33, 22.22, 2)` retourne `xs:string "33.33S 22.22E"`
  - **`altova:format-geolocation`**`(33.33, -22.22, 1)` retourne `xs:string "33°19'48.00"S 22° 13'12.00"E"`

- *Formats de chaîne de sortie de géolocalisation:*

  La latitude et longitude fournies sont formatées dans un des formats de sortie indiqués ci-dessous. Le format désiré est défini par son ID d'entier (`1 à 4`). Les valeurs de latitude vont de `+90` à `-90` (`N` à `S`). Les valeurs de longitude vont de `+180` à `-180` (`E` à `O`).

1
Degrés, minutes, secondes décimales, avec orientation suffixée (`N/S, E/O`) **`D°M'S.SS"N/S D°M'S.SS"E/W`** *Exemple :* **`33°55'11.11"N 22°44'66.66"W`**

2
Degrés décimaux, avec orientation suffixée (`N/S, E/O`) **`D.DDN/S D.DDE/W`** *Exemple :* **`33.33N 22.22W`**

3
Degrés, minutes, secondes décimales, avec signe préfixé (`+/-`); le signe plus (**`N/E`**) est optionnel **`+/-D°M'S.SS" +/-D°M'S.SS"`** *Exemple :* **`33°55'11.11" -22°44'66.66"`**

4
Degrés décimaux, avec signe préfixé (`+/-`); le signe plus (**`N/E`**) est optionnel **`+/-D.DD +/-D.DD`** *Exemple :* **`33.33 -22.22`**

- *Attribut Altova Exif : Géolocalisation*

  La machine Altova XPath/XQuery génère l'attribut **`Geolocation`** personnalisable depuis les onglets standard de métadonnées Exif. **`Geolocation`** est une concaténation de quatre onglets Exif : `GPSLatitude`, `GPSLatitudeRef`, `GPSLongitude`, `GPSLongitudeRef`, avec des unités ajoutées (voir table ci-dessous).

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation

33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

parse-geolocation [altova:]

altova:parse-geolocation(GeolocationInputString *as xs:string*) **asxs:decimal+** XP3.1 XQ3.1
Parse l'argument `GeolocationInputString` fourni et retourne la latitude et longitude de géolocalisation (dans cet ordre) en séquence deux items décimaux `xs:decimal`. Les formats dans lesquels la chaîne d'entrée de géolocalisation peut être fournie sont recensés ci-dessous.

Note : La fonction image-exif-data 525 et l'attribut @Geolocation 525 de métadonnées Exif peuvent être utilisés pour fournir la chaîne d'entrée de géolocalisation (*voir exemple ci-dessous*).

Exemples

- **altova:parse-geolocation**("33.33 -22.22") retourne la séquence de deux `xs:decimals` (33.33, 22.22)
- **altova:parse-geolocation**("48°51'29.6""N 24°17'40.2""") retourne la séquence de deux `xs:decimals` (48.8582222222222, 24.2945)
- **altova:parse-geolocation**('48°51''29.6"N 24°17''40.2"') retourne la séquence de deux `xs:decimals` (48.8582222222222, 24.2945)
- **altova:parse-geolocation**(**image-exif-data**(//MyImages/Image20141130.01)/**@Geolocation**)retourne une séquence de deux `xs:decimals`

Formats de string d'entrée de géolocalisation :

Le string d'entrée de géolocalisation doit contenir la latitude et la longitude (dans cet ordre) séparées par un espace. Les strings peuvent tous présenter les formats suivants. Les combinaisons sont permises. La latitude peut donc être dans un format et la longitude dans un autre. Les valeurs de latitude varient de `+90` à `-90` (`N` à `S`). Les valeurs de longitude varient de `+180` à `-180` (`E` à `W`).

Note : L'utilisation de guillemets simples ou doubles pour la délimitation des arguments de string entraînera une non-concordance avec l'utilisation de guillemets simples ou doubles pour indiquer, respectivement les valeurs de minutes et de secondes. Dans ces cas, les guillemets utilisés pour indiquer les minutes et les secondes doivent être échappés en les doublant. Dans les exemples présentés dans cette section, les guillemets utilisés pour délimiter les strings d'entrée sont marqués en jaune (") alors que les indicateurs d'unité échappés sont marqués en bleu ("").

- Degrés, minutes, secondes décimales, avec orientation suffixée (`N/S`, `E/W`)
  D°M'S.SS"N/S D°M'S.SS"W/E
  Exemple : **33°55'11.11"N 22°44'55.25"W**

- Degrés, minutes, secondes décimales, avec signe préfixé (`+/-`) ; le signe plus pour (**N/E**) est optionnel
  +/-D°M'S.SS" +/-D°M'S.SS"
  Exemple : **33°55'11.11" -22°44'55.25"**

- Degrés, minutes décimales, avec orientation suffixée (N/S, E/W)
  D°M.MM'N/S D°M.MM'W/E
  Exemple : **33°55.55'N 22°44.44'W**

- Degrés, minutes décimales, avec signe préfixé (+/-) ; le signe plus pour (**N/E**) est optionnel
  +/-D°M.MM' +/-D°M.MM'
  Exemple : **+33°55.55' -22°44.44'**

- Degrés décimaux, avec orientation suffixée (N/S, E/W)
  D.DDN/S D.DDW/E
  Exemple : **33.33N 22.22W**

- Degrés décimaux, avec signe préfixé (+/-) ; le signe plus pour (**N/S E/W**) est optionnel
  +/-D.DD +/-D.DD
  Exemple : **33.33 -22.22**

Exemples de combinaisons de format :
33.33N -22°44'55.25"
33.33 22°44'55.25"W
33.33 22.45

Attribut Altova Exif : Géolocalisation

La machine Altova XPath/XQuery génère l'attribut **Geolocation** personnalisable depuis les onglets standard de métadonnées Exif. **Geolocation** est une concaténation de quatre onglets Exif : GPSLatitude, GPSLatitudeRef, GPSLongitude, GPSLongitudeRef, avec des unités ajoutées (voir table ci-dessous).

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(GeolocationInputString-1 *as xs:string*, **GeolocationInputString-2** *as xs:string*) **asxs:decimal** XP3.1 XQ3.1
Calcule la distance entre deux géolocalisations en kilomètres. Les formats dans lesquels une chaîne d'entrée de géolocalisation peut être fournie sont recensés ci-dessous. Les valeurs de latitude vont de +90 à -90 (N à S). Les valeurs de longitude vont de +180 à -180 (E à O).

Note : La fonction image-exif-data 525 et l'attribut @Geolocation 525 des métadonnées d'Exif peuvent être utilisés pour fournir les chaînes d'entrée de géolocalisation.

Exemples

- **altova:geolocation-distance-km**("33.33 -22.22", "48°51'29.6""N 24°17'40.2""")
  retourne xs:decimal 4183.08132372392

Formats de string d'entrée de géolocalisation :

Le string d'entrée de géolocalisation doit contenir la latitude et la longitude (dans cet ordre) séparées par un espace. Les strings peuvent tous présenter les formats suivants. Les combinaisons sont permises. La latitude peut donc être dans un format et la longitude dans un autre. Les valeurs de latitude varient de `+90` à `-90` (`N` à `S`). Les valeurs de longitude varient de `+180` à `-180` (`E` à `W`).

Note : L'utilisation de guillemets simples ou doubles pour la délimitation des arguments de string entraînera une non-concordance avec l'utilisation de guillemets simples ou doubles pour indiquer, respectivement les valeurs de minutes et de secondes. Dans ces cas, les guillemets utilisés pour indiquer les minutes et les secondes doivent être échappés en les doublant. Dans les exemples présentés dans cette section, les guillemets utilisés pour délimiter les strings d'entrée sont marqués en jaune (**''**) alors que les indicateurs d'unité échappés sont marqués en bleu (**""**).

- Degrés, minutes, secondes décimales, avec orientation suffixée (`N/S`, `E/W`)
  `D°M'S.SS"N/S  D°M'S.SS"W/E`
  Exemple : **`33°55'11.11"N  22°44'55.25"W`**

- Degrés, minutes, secondes décimales, avec signe préfixé (`+/-`) ; le signe plus pour (**`N/E`**) est optionnel
  `+/-D°M'S.SS"  +/-D°M'S.SS"`
  Exemple : **`33°55'11.11"  -22°44'55.25"`**

- Degrés, minutes décimales, avec orientation suffixée (`N/S`, `E/W`)
  `D°M.MM'N/S  D°M.MM'W/E`
  Exemple : **`33°55.55'N  22°44.44'W`**

- Degrés, minutes décimales, avec signe préfixé (`+/-`) ; le signe plus pour (**`N/E`**) est optionnel
  `+/-D°M.MM'  +/-D°M.MM'`
  Exemple : **`+33°55.55'  -22°44.44'`**

- Degrés décimaux, avec orientation suffixée (`N/S`, `E/W`)
  `D.DDN/S  D.DDW/E`
  Exemple : **`33.33N  22.22W`**

- Degrés décimaux, avec signe préfixé (`+/-`) ; le signe plus pour (**`N/S E/W`**) est optionnel
  `+/-D.DD  +/-D.DD`
  Exemple : **`33.33  -22.22`**

Exemples de combinaisons de format :
`33.33N  -22°44'55.25"`
`33.33  22°44'55.25"W`
`33.33  22.45`

Attribut Altova Exif : Géolocalisation

La machine Altova XPath/XQuery génère l'attribut **`Geolocation`** personnalisable depuis les onglets standard de métadonnées Exif. **`Geolocation`** est une concaténation de quatre onglets Exif : `GPSLatitude`, `GPSLatitudeRef`, `GPSLongitude`, `GPSLongitudeRef`, avec des unités ajoutées (voir table ci-dessous).

GPSLatitude	GPSLatitudeRe	GPSLongitude	GPSLongitudeRe	Geolocation

	f		f	
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

geolocation-distance-mi [altova:]

altova:geolocation-distance-mi(GeolocationInputString-1 *as xs:string*, **GeolocationInputString-2** *as xs:string*) **asxs:decimal** XP3.1 XQ3.1

Calcule la distance entre deux géolocalisations en miles. Les formats dans lesquels une chaîne d'entrée de géolocalisation peut être fournie sont recensés ci-dessous. Les valeurs de latitude vont de +90 à -90 (N à S). Les valeurs de longitude vont de +180 à -180 (E à O).

Note : La fonction image-exif-data 525 et l'attribut @Geolocation 525 des métadonnées d'Exif peuvent être utilisés pour fournir les chaînes d'entrée de géolocalisation.

Exemples

- **altova:geolocation-distance-mi**("33.33 -22.22", "48°51'29.6""N 24°17'40.2""") retourne xs:decimal 2599.40652340653

Formats de string d'entrée de géolocalisation :

Le string d'entrée de géolocalisation doit contenir la latitude et la longitude (dans cet ordre) séparées par un espace. Les strings peuvent tous présenter les formats suivants. Les combinaisons sont permises. La latitude peut donc être dans un format et la longitude dans un autre. Les valeurs de latitude varient de +90 à -90 (N à S). Les valeurs de longitude varient de +180 à -180 (E à W).

Note : L'utilisation de guillemets simples ou doubles pour la délimitation des arguments de string entraînera une non-concordance avec l'utilisation de guillemets simples ou doubles pour indiquer, respectivement les valeurs de minutes et de secondes. Dans ces cas, les guillemets utilisés pour indiquer les minutes et les secondes doivent être échappés en les doublant. Dans les exemples présentés dans cette section, les guillemets utilisés pour délimiter les strings d'entrée sont marqués en jaune (") alors que les indicateurs d'unité échappés sont marqués en bleu ("").

- Degrés, minutes, secondes décimales, avec orientation suffixée (N/S, E/W)
  D°M'S.SS"N/S D°M'S.SS"W/E
  Exemple : **33°55'11.11"N 22°44'55.25"W**

- Degrés, minutes, secondes décimales, avec signe préfixé (+/-) ; le signe plus pour (**N/E**) est optionnel
  +/-D°M'S.SS" +/-D°M'S.SS"
  Exemple : **33°55'11.11" -22°44'55.25"**

- Degrés, minutes décimales, avec orientation suffixée (N/S, E/W)
  D°M.MM'N/S D°M.MM'W/E
  Exemple : **33°55.55'N 22°44.44'W**

- Degrés, minutes décimales, avec signe préfixé (+/-) ; le signe plus pour (**N/E**) est optionnel

+/-D°M.MM' +/-D°M.MM'
Exemple : **+33°55.55' -22°44.44'**

- Degrés décimaux, avec orientation suffixée (N/S, E/W)
  D.DDN/S D.DDW/E
  Exemple : **33.33N 22.22W**

- Degrés décimaux, avec signe préfixé (+/-) ; le signe plus pour (**N/S E/W**) est optionnel
  +/-D.DD +/-D.DD
  Exemple : **33.33 -22.22**

Exemples de combinaisons de format :
33.33N -22°44'55.25"
33.33 22°44'55.25"W
33.33 22.45

Attribut Altova Exif : Géolocalisation

La machine Altova XPath/XQuery génère l'attribut **Geolocation** personnalisable depuis les onglets standard de métadonnées Exif. **Geolocation** est une concaténation de quatre onglets Exif : GPSLatitude, GPSLatitudeRef, GPSLongitude, GPSLongitudeRef, avec des unités ajoutées (voir table ci-dessous).

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(Geolocation *as xs:string*, **((PolygonPoint** *as xs:string*)+)) **asxs:boolean** XP3.1 XQ3.1
Détermine si **Geolocation** (le premier argument) se trouve dans l'espace polygonal décrit par les arguments **PolygonPoint**. Si les arguments PolygonPoint ne forment pas une figure fermée (formée lorsque le premier point et le dernier point sont identiques), alors le premier point est implicitement ajouté en tant que le dernier point afin de pouvoir clore la figure. Tous les arguments (Geolocation et PolygonPoint+) sont donnés par chaînes d'entrées de géolocalisation (*formats recensés ci-dessous*). Si l'argument Geolocation se trouve dans l'espace polygonal, la fonction retourne true(); sinon, elle retourne false(). Les valeurs de latitude vont de +90 à -90 (N à S). Les valeurs de longitude vont de +180 à -180 (E à O).

Note : La fonction image-exif-data 525 et l'attribut @Geolocation 525 de métadonnées d'Exif peut être utilisée pour fournir les chaînes d'entrée de géolocalisation.

Exemples

- **altova:geolocation-within-polygon(**"33 -22", ("58 -32", "-78 -55", "48 24", "58 -32"))** retourne true()

- `altova:geolocation-within-polygon("33 -22", ("58 -32", "-78 -55", "48 24"))` retourne `true()`
- `altova:geolocation-within-polygon("33 -22", ("58 -32", "-78 -55", "48°51'29.6""N 24°17'40.2""))` retourne `true()`

Formats de string d'entrée de géolocalisation :

Le string d'entrée de géolocalisation doit contenir la latitude et la longitude (dans cet ordre) séparées par un espace. Les strings peuvent tous présenter les formats suivants. Les combinaisons sont permises. La latitude peut donc être dans un format et la longitude dans un autre. Les valeurs de latitude varient de `+90` à `-90` (`N` à `S`). Les valeurs de longitude varient de `+180` à `-180` (`E` à `W`).

Note : L'utilisation de guillemets simples ou doubles pour la délimitation des arguments de string entraînera une non-concordance avec l'utilisation de guillemets simples ou doubles pour indiquer, respectivement les valeurs de minutes et de secondes. Dans ces cas, les guillemets utilisés pour indiquer les minutes et les secondes doivent être échappés en les doublant. Dans les exemples présentés dans cette section, les guillemets utilisés pour délimiter les strings d'entrée sont marqués en jaune (") alors que les indicateurs d'unité échappés sont marqués en bleu ("").

- Degrés, minutes, secondes décimales, avec orientation suffixée (`N/S`, `E/W`)
  `D°M'S.SS"N/S D°M'S.SS"W/E`
  Exemple : **`33°55'11.11"N 22°44'55.25"W`**

- Degrés, minutes, secondes décimales, avec signe préfixé (`+/-`) ; le signe plus pour (**`N/E`**) est optionnel
  `+/-D°M'S.SS" +/-D°M'S.SS"`
  Exemple : **`33°55'11.11" -22°44'55.25"`**

- Degrés, minutes décimales, avec orientation suffixée (`N/S`, `E/W`)
  `D°M.MM'N/S D°M.MM'W/E`
  Exemple : **`33°55.55'N 22°44.44'W`**

- Degrés, minutes décimales, avec signe préfixé (`+/-`) ; le signe plus pour (**`N/E`**) est optionnel
  `+/-D°M.MM' +/-D°M.MM'`
  Exemple : **`+33°55.55' -22°44.44'`**

- Degrés décimaux, avec orientation suffixée (`N/S`, `E/W`)
  `D.DDN/S D.DDW/E`
  Exemple : **`33.33N 22.22W`**

- Degrés décimaux, avec signe préfixé (`+/-`) ; le signe plus pour (**`N/S E/W`**) est optionnel
  `+/-D.DD +/-D.DD`
  Exemple : **`33.33 -22.22`**

Exemples de combinaisons de format :
`33.33N -22°44'55.25"`
`33.33 22°44'55.25"W`
`33.33 22.45`

Attribut Altova Exif : Géolocalisation

La machine Altova XPath/XQuery génère l'attribut **`Geolocation`** personnalisable depuis les onglets

standard de métadonnées Exif. **Geolocation** est une concaténation de quatre onglets Exif : GPSLatitude, GPSLatitudeRef, GPSLongitude, GPSLongitudeRef, avec des unités ajoutées (voir table ci-dessous).

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(Geolocation *as xs:string*, **RectCorner-1** *as xs:string*, **RectCorner-2** *as xs:string*) **as xs:boolean** XP3.1 XQ3.1

Détermine si **Geolocation** (le premier argument) se trouve dans le rectangle défini par le second et le troisième argument, **RectCorner-1** et **RectCorner-2**, qui spécifient les coins opposés du rectangle. Tous les arguments (Geolocation, **RectCorner-1** et **RectCorner-2**) sont indiqués par des chaînes d'entrées de géolocalisation (*formats recensés ci-dessous*). Si l'argument Geolocation se trouve dans le rectangle, la fonction retourne true(); sinon, elle retourne false(). Les valeurs de latitude vont de +90 à -90 (N à S). Les valeurs de longitude vont de +180 à -180 (E à O).

Note : La fonction image-exif-data 525 et l'attribut @Geolocation 525 de métadonnées Exif peuvent être utilisés pour fournir les chaînes d'entrée de géolocalisation.

Exemples

- **altova:geolocation-within-rectangle(**"33 -22", "58 -32", "-48 24") retourne true()
- **altova:geolocation-within-rectangle(**"33 -22", "58 -32", "48 24") retourne false()
- **altova:geolocation-within-rectangle(**"33 -22", "58 -32", "48°51'29.6""S 24° 17'40.2""") retourne true()

Formats de string d'entrée de géolocalisation :

Le string d'entrée de géolocalisation doit contenir la latitude et la longitude (dans cet ordre) séparées par un espace. Les strings peuvent tous présenter les formats suivants. Les combinaisons sont permises. La latitude peut donc être dans un format et la longitude dans un autre. Les valeurs de latitude varient de +90 à -90 (N à S). Les valeurs de longitude varient de +180 à -180 (E à W).

Note : L'utilisation de guillemets simples ou doubles pour la délimitation des arguments de string entraînera une non-concordance avec l'utilisation de guillemets simples ou doubles pour indiquer, respectivement les valeurs de minutes et de secondes. Dans ces cas, les guillemets utilisés pour indiquer les minutes et les secondes doivent être échappés en les doublant. Dans les exemples présentés dans cette section, les guillemets utilisés pour délimiter les strings d'entrée sont marqués en jaune (") alors que les indicateurs d'unité échappés sont marqués en bleu ("").

- Degrés, minutes, secondes décimales, avec orientation suffixée (N/S, E/W)
  D°M'S.SS"N/S D°M'S.SS"W/E
  Exemple : **33°55'11.11"N 22°44'55.25"W**

- Degrés, minutes, secondes décimales, avec signe préfixé (`+/-`) ; le signe plus pour (**`N/E`**) est optionnel
  `+/-D°M'S.SS"  +/-D°M'S.SS"`
  Exemple : **`33°55'11.11"  -22°44'55.25"`**

- Degrés, minutes décimales, avec orientation suffixée (`N/S`, `E/W`)
  `D°M.MM'N/S  D°M.MM'W/E`
  Exemple : **`33°55.55'N  22°44.44'W`**

- Degrés, minutes décimales, avec signe préfixé (`+/-`) ; le signe plus pour (**`N/E`**) est optionnel
  `+/-D°M.MM'  +/-D°M.MM'`
  Exemple : **`+33°55.55'  -22°44.44'`**

- Degrés décimaux, avec orientation suffixée (`N/S`, `E/W`)
  `D.DDN/S  D.DDW/E`
  Exemple : **`33.33N  22.22W`**

- Degrés décimaux, avec signe préfixé (`+/-`) ; le signe plus pour (**`N/S E/W`**) est optionnel
  `+/-D.DD  +/-D.DD`
  Exemple : **`33.33  -22.22`**

Exemples de combinaisons de format :
`33.33N  -22°44'55.25"`
`33.33  22°44'55.25"W`
`33.33  22.45`

Attribut Altova Exif : Géolocalisation

La machine Altova XPath/XQuery génère l'attribut **`Geolocation`** personnalisable depuis les onglets standard de métadonnées Exif. **`Geolocation`** est une concaténation de quatre onglets Exif : `GPSLatitude`, `GPSLatitudeRef`, `GPSLongitude`, `GPSLongitudeRef`, avec des unités ajoutées (voir table ci-dessous).

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

[Top 516]

12.2.2.1.4 Fonctions XPath/XQuery : Relatives aux images

Les fonctions d'extension XPath/XQuery relatives à l'image suivantes sont prises en charge dans la version actuelle de MapForce et peuvent être utilisées dans (i) des expressions XPath dans un contexte XSLT, ou dans (ii) des expressions XQuery dans un document XQuery.

Note concernant le nommage de fonctions et de l'applicabilité de la langue

Les fonctions d'extension Altova peuvent être utilisées dans les expressions XPath/XQuery. Elles fournissent des fonctions supplémentaires aux fonctions d'ores et déjà disponibles dans la librairie standard des fonctions XPath, XQuery et XSLT. Les fonctions d'extension Altova se trouvent dans **l'espace de nom des fonctions d'extension Altova, `http://www.altova.com/xslt-extensions`**, et sont indiquées dans cette section par le préfixe **`altova:`**, qui est présupposé être lié à cet espace de nom. Veuillez noter que, en ce qui concerne les versions futures de votre produit, la prise en charge d'une fonction peut être interrompue et le comportement de certaines fonctions peut changer. Veuillez consulter la documentation lors des publications à venir pour plus d'informations concernant la prise en charge des fonctions d'extension Altova de cette version.

Fonctions XPath (utilisées dans les expressions XPath dans XSLT) :	XP1 XP2 XP3.1
Fonctions XSLT (utilisées dans les expressions XPath dans XSLT) :	XSLT1 XSLT2 XSLT3
Fonctions XQuery (utilisées dans les expressions XQuery dans XQuery) :	XQ1 XQ3.1

suggested-image-file-extension [altova:]

`altova:suggested-image-file-extension(Base64String` *`as string`***`) asstring?`** XP3.1 XQ3.1
Prend le code Base64 d'un fichier d'image en tant que son argument et retourne l'extension de fichier de l'image comme enregistré dans le codage Base64 de l'image. La valeur retournée est une suggestion basée sur l'information du type d'image disponible dans le codage. Si cette information n'est pas disponible, une chaîne vide est retournée. Cette fonction est utile si vous souhaitez enregistrer une image Base64 en tant que fichier et que vous souhaitez extraire dynamiquement une extension de fichier appropriée.

Exemples

- **`altova:suggested-image-file-extension`**`(/MyImages/MobilePhone/Image20141130.01)` retourne `'jpg'`
- **`altova:suggested-image-file-extension`**`($XML1/Staff/Person/@photo)` retourne `''`

Dans les exemples ci-dessus, les nœuds fournis en tant qu'arguments de la fonction sont assumés contenir une image codée Base64. Le premier exemple extrait `jpg` en tant que type et extension de fichier. Dans le second exemple, le codage Base64 soumis ne fournit pas une information de fichier d'extension utile.

image-exif-data [altova:]

`altova:image-exif-data(Base64BinaryString` *`as string`***`) aselement?`** XP3.1 XQ3.1
Prend une image codée Base64 en tant que son argument et retourne un élément appelé **`Exif`** qui contient les métadonnées Exif de l'image. Celles-ci sont créées en tant que paires attribute-value pairs de l'élément `Exif`. Les noms d'attribut sont les onglets de données Exif trouvés dans le codage Base64. La liste des onglets des spécifications Exif est indiquée ci-dessous. Si un onglet spécifique à un distributeur est présent dans les données Exif, cet onglet et sa valeur seront aussi retournés en tant que paire attribute-value. Outre les onglets de métadonnées Exif standard (*voir la liste ci-dessous*), des paires attribute-value spécifiques à Altova sont également générées. Ces attributs Exif Altova sont recensés ci-

dessous.

Exemples

- Pour accéder à n'importe quel attribut, utiliser la fonction comme suit :
  `image-exif-data(//MyImages/Image20141130.01)/@GPSLatitude`
  `image-exif-data(//MyImages/Image20141130.01)/@Geolocation`
- Pour accéder à tous les attributs, utiliser la fonction comme suit:
  `image-exif-data(//MyImages/Image20141130.01)/@*`
- Pour accéder au nom de tous les attributs, utiliser l'expression suivante :
  `for $i in image-exif-data(//MyImages/Image20141130.01)/@* return name($i)`
  Cela est utile pour trouver les noms des attributs retournés par la fonction.

Attribut Altova Exif : Géolocalisation

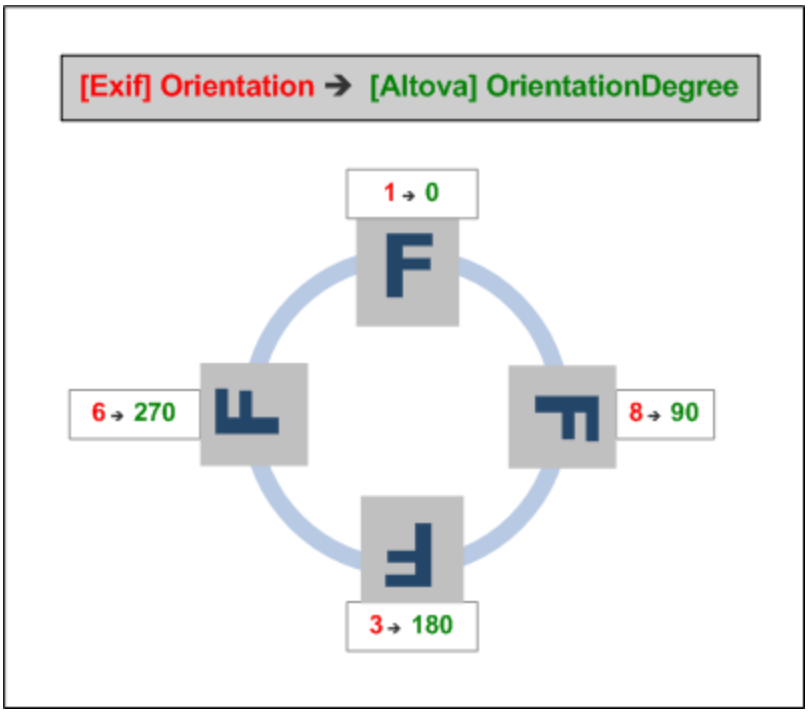
La machine Altova XPath/XQuery génère l'attribut **Geolocation** personnalisable depuis les onglets standard de métadonnées Exif. **Geolocation** est une concaténation de quatre onglets Exif : `GPSLatitude`, `GPSLatitudeRef`, `GPSLongitude`, `GPSLongitudeRef`, avec des unités ajoutées (voir table ci-dessous).

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

Altova Exif Attribute: OrientationDegree

La machine Altova XPath/XQuery génère l'attribut personnalisé **OrientationDegree** à partir de l'onglet de métadonnées Exif **Orientation**.

OrientationDegree traduit l'onglet standard Exif `Orientation` à partir d'une valeur d'entier (`1`, `8`, `3`, ou `6`) aux valeurs de degrés respectives de chacun (`0`, `90`, `180`, `270`), tel que montré dans la figure ci-dessous. Veuillez noter qu'il n'y a pas de traductions de la valeur `Orientation` de `2`, `4`, `5`, `7`. (Ces orientations sont obtenus en basculant l'image `1` à travers son centre axial vertical pour obtenir l'image avec une valeur de `2`, puis en pivotant cette image par sauts de 90° dans le sens des aiguilles d'une montre pour obtenir les valeurs de `7`, `4`, et `5`, respectivement).



- *Listing of standard Exif meta tags*
  - 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

[Top 525]

12.2.2.1.5 Fonctions XPath/XQuery : Numérique

Les fonctions d'extension numériques d'Altova peuvent être utilisées dans des expressions XPath et XQuery et proposent des fonctions supplémentaires pour le traitement des données. Les fonctions dans cette section peuvent être utilisée avec les moteurs **XPath 3.0** et **XQuery 3.0** d'Altova. Ils sont disponibles dans des contextes XPath/XQuery.

Note concernant le nommage de fonctions et de l'applicabilité de la langue

Les fonctions d'extension Altova peuvent être utilisées dans les expressions XPath/XQuery. Elles fournissent des fonctions supplémentaires aux fonctions d'ores et déjà disponibles dans la librairie standard des

fonctions XPath, XQuery et XSLT. Les fonctions d'extension Altova se trouvent dans **l'espace de nom des fonctions d'extension Altova, `http://www.altova.com/xslt-extensions`**, et sont indiquées dans cette section par le préfixe **`altova:`**, qui est présupposé être lié à cet espace de nom. Veuillez noter que, en ce qui concerne les versions futures de votre produit, la prise en charge d'une fonction peut être interrompue et le comportement de certaines fonctions peut changer. Veuillez consulter la documentation lors des publications à venir pour plus d'informations concernant la prise en charge des fonctions d'extension Altova de cette version.

Fonctions XPath (utilisées dans les expressions XPath dans XSLT) :	XP1 XP2 XP3.1
Fonctions XSLT (utilisées dans les expressions XPath dans XSLT) :	XSLT1 XSLT2 XSLT3
Fonctions XQuery (utilisées dans les expressions XQuery dans XQuery) :	XQ1 XQ3.1

Fonctions de numérotation automatique

generate-auto-number [altova:]

`altova:generate-auto-number(ID` *as xs:string*`, StartsWith` *as xs:double*`, Increment` *as xs:double*`, ResetOnChange` *as xs:string*`)` as `xs:integer` XP1 XP2 XQ1 XP3.1 XQ3.1

Génère un numéro à chaque fois que la fonction est appelée. Le premier numéro, qui est généré la première fois que la fonction est appelée, est spécifié par l'argument `StartsWith`. Chaque appel subséquent vers la fonction génère un nouveau numéro, ce numéro est augmenté au-dessus du numéro précédemment généré par la valeur spécifiée dans l'argument `Increment`. En effet, la fonction `altova:generate-auto-number` crée un compteur comportant un nom spécifié par l'argument `ID`, et dont le compteur est augmenté à chaque fois que la fonction est appelée. Si la valeur de l'argument `ResetOnChange` change de celle de l'appel de fonction précédent, la valeur du numéro à générer est réinitialisée à la valeur `StartsWith`. La numérotation automatique peut être réinitialisée en utilisant la fonction `altova:reset-auto-number`.

Exemples

- **`altova:generate-auto-number`**`("ChapterNumber", 1, 1, "SomeString")` retournera un nombre à chaque fois que la fonction est appelée, en commençant avec `1`, et en augmentant de `1` avec chaque appel de la fonction. Tant que le quatrième argument demeure `"SomeString"` dans chaque appel subséquent, l'augmentation se poursuivra. Lorsque la valeur du quatrième argument change, le compteur (appelé `ChapterNumber`) sera réinitialisé à 1. La valeur de `ChapterNumber` peut aussi être réinitialisée par un appel de la fonction `altova:reset-auto-number` comme ceci : `altova:reset-auto-number("ChapterNumber")`.

reset-auto-number [altova:]

`altova:reset-auto-number(ID` *as xs:string*`)` XP1 XP2 XQ1 XP3.1 XQ3.1

Cette fonction réinitialise le numéro du compteur de numérotation automatique nommé dans l'argument `ID`. Le numéro est réinitialisé au numéro spécifié par l'argument `StartsWith` de la fonction `altova:generate-auto-number` qui a créé le compteur nommé dans l'argument `ID`.

Exemples

- **`altova:reset-auto-number`**`("ChapterNumber")` réinitialise le numéro du compteur de

numérotation automatique nommé `ChapterNumber` qui a été créé par la fonction `altova:generate-auto-number`. Le numéro est réinitialisé à la valeur de l'argument `StartsWith` de la fonction `altova:generate-auto-number` qui a créé `ChapterNumber`.

[Top 530]

Fonctions numériques

hex-string-to-integer [altova:]

`altova:hex-string-to-integer(HexString` *`as xs:string`***`) asxs:integer`** **XP3.1** **XQ3.1**

Prend un argument de chaîne qui est l'équivalent Base-16 d'un entier dans le système décimal (Base-10), et retourne l'entier décimal.

Exemples

- **`altova:hex-string-to-integer`**`('1')` retourne `1`
- **`altova:hex-string-to-integer`**`('9')` retourne `9`
- **`altova:hex-string-to-integer`**`('A')` retourne `10`
- **`altova:hex-string-to-integer`**`('B')` retourne `11`
- **`altova:hex-string-to-integer`**`('F')` retourne `15`
- **`altova:hex-string-to-integer`**`('G')` retourne une erreur
- **`altova:hex-string-to-integer`**`('10')` retourne `16`
- **`altova:hex-string-to-integer`**`('01')` retourne `1`
- **`altova:hex-string-to-integer`**`('20')` retourne `32`
- **`altova:hex-string-to-integer`**`('21')` retourne `33`
- **`altova:hex-string-to-integer`**`('5A')` retourne `90`
- **`altova:hex-string-to-integer`**`('USA')` retourne une erreur

integer-to-hex-string [altova:]

`altova:integer-to-hex-string(Integer` *`as xs:integer`***`) asxs:string`** **XP3.1** **XQ3.1**

Prend un argument d'entier et retourne son équivalent de Base-16 en tant que chaîne.

Exemples

- **`altova:integer-to-hex-string`**`(1)` retourne `'1'`
- **`altova:integer-to-hex-string`**`(9)` retourne `'9'`
- **`altova:integer-to-hex-string`**`(10)` retourne `'A'`
- **`altova:integer-to-hex-string`**`(11)` retourne `'B'`
- **`altova:integer-to-hex-string`**`(15)` retourne `'F'`
- **`altova:integer-to-hex-string`**`(16)` retourne `'10'`
- **`altova:integer-to-hex-string`**`(32)` retourne `'20'`
- **`altova:integer-to-hex-string`**`(33)` retourne `'21'`
- **`altova:integer-to-hex-string`**`(90)` retourne `'5A'`

[Top 530]

Fonctions de formatage de numéro

[Top 530]

12.2.2.1.6 Fonctions XPath/XQuery : Schéma

Les fonctions d'extension Altova recensées ci-dessous retournent l'information de schéma. Ci-dessous, vous trouverez les descriptions des fonctions, ainsi que des (i) exemples et (ii) une liste des composants de schéma et de leurs propriétés respectives. Elles peuvent être utilisées avec les moteurs **XPath 3.0** et **XQuery 3.0** d'Altova et sont disponibles dans des contextes XPath/XQuery.

Information de schéma depuis les documents de schéma
La fonction **altova:schema** détient deux arguments : un avec zéro arguments et l'autre avec deux arguments. La fonction à zéro argument retourne l'ensemble du schéma. Ensuite, à partir de là, vous pouvez naviguer dans le schéma pour localiser les composants de schéma que vous souhaitez. La fonction à deux arguments retourne un type de composant spécifique qui est identifié par son QName. Dans les deux cas, la valeur de retour est un fonction. Pour naviguer dans le composant retourné, vous devez sélectionner une propriété de ce composant spécifique. Si la propriété est un item non atomique (c'est à dire, s'il s'agit d'un composant), vous pouvez aller plus loin en choisissant une propriété de ce composant. Si la propriété sélectionnée est un item atomique, la valeur de l'item est retournée et vous ne pouvez pas aller plus loin.

Note : Dans des expressions XPath, le schéma doit avoir été importé dans l'environnement de traitement, par exemple, dans XSLT avec l'instruction **xslt:import-schema**. Dans des expressions XQuery, le schéma doit être importé explicitement utilisant un schema import.

Information de schéma depuis les nœuds XML
La fonction **altova:type** soumet le nœud à un document XML et retourne l'information de type du nœud depuis le PSVI.

Note concernant le nommage de fonctions et de l'applicabilité de la langue

Les fonctions d'extension Altova peuvent être utilisées dans les expressions XPath/XQuery. Elles fournissent des fonctions supplémentaires aux fonctions d'ores et déjà disponibles dans la librairie standard des fonctions XPath, XQuery et XSLT. Les fonctions d'extension Altova se trouvent dans **l'espace de nom des fonctions d'extension Altova, http://www.altova.com/xslt-extensions**, et sont indiquées dans cette section par le préfixe **altova:**, qui est présupposé être lié à cet espace de nom. Veuillez noter que, en ce qui concerne les versions futures de votre produit, la prise en charge d'une fonction peut être interrompue et le comportement de certaines fonctions peut changer. Veuillez consulter la documentation lors des publications à venir pour plus d'informations concernant la prise en charge des fonctions d'extension Altova de cette version.

Fonctions XPath (utilisées dans les expressions XPath dans XSLT) :	XP1 XP2 XP3.1
Fonctions XSLT (utilisées dans les expressions XPath dans XSLT) :	XSLT1 XSLT2 XSLT3
Fonctions XQuery (utilisées dans les expressions XQuery	XQ1 XQ3.1

dans XQuery) :	

▼ Schéma (zéro arguments)

altova:schema() as (function(xs:string) as item()*)? XP3.1 XQ3.1
Retourne le composant de **schema** en entier. Vous pouvez donc aller plus loin dans le composant de **schema** en sélectionnant une des propriétés du composant de **schema**.

- Si cette propriété est un composant, vous pouvez aller encore plus en loin en sélectionnant une des propriétés de ce composant. Vous pouvez renouveler cette étape en allant plus loin dans le schéma.
- Si le composant est une valeur atomique, celle-ci sera retournée et vous ne pourrez pas aller plus loin.

Les propriétés du composant de **schema** sont :

```
"type definitions"
"attribute declarations"
"element declarations"
"attribute group definitions"
"model group definitions"
"notation declarations"
"identity-constraint definitions"
```

Les propriétés de tous les types de composant (à part **schema**) sont regroupées ci-dessous.

Note: Dans des expressions XQuery, le schéma doit être importé explicitement. Dans des expressions XPath, le schéma doit avoir été importé dans l'environnement de traitement, par exemple dans XSLT avec l'instruction **xslt:import**.

▫ *Exemples*

- `import schema "" at "C:\Test\ExpReport.xsd"; for $typedef in altova:schema() ("type definitions") return $typedef ("name")` retourne les noms de tous les Types simples ou Types complexes dans le schéma

- `import schema "" at "C:\Test\ExpReport.xsd"; altova:schema() ("type definitions")[1]("name")` retourne le nom du premier de tous les Types simples ou Types complexes dans le schéma

Composants et leurs propriétés

▫ Assertion

Nom de propriété	Type de propriété	Valeur de propriété

kind	string	"Assertion"
test	XPath Property Record	

Déclaration d'attribut

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Attribute Declaration"
name	string	Local name of the attribute
target namespace	string	Namespace URI of the attribute
type definition	Type simple or Type complexe	
scope	A function with properties ("class":"Scope", "variety": "global" or "local", "parent": the containing Type complexe or Attribute Group)	
value constraint	If present, a function with properties ("class": "Value Constraint", "variety": "fixed" or "default", "value": atomic value, "lexical form": string. Note that the "value" property is not available for namespace-sensitive types	
inheritable	boolean	

Déclaration de groupe d'attribut

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Attribute Group Definition"
name	string	Local name of the attribute group
target namespace	string	Namespace URI of the attribute group
attribute uses	Sequence of (Attribute Use)	
attribute wildcard	Optional Attribute Wildcard	

Utilisation d'attribut

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Attribute Use"
required	boolean	true if the attribute is required, false if optional
value constraint	See Attribute Declaration	
inheritable	boolean	

Caractère générique

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Wildcard"
namespace constraint	function with properties ("class": "Namespace Constraint", "variety": "any"|"enumeration"|"not", "namespaces": sequence of xs:anyURI, "disallowed names": list containing QNames and/or the strings "defined" and "definedSiblings"	
process contents	string ("strict"|"lax"|"skip")	

Type complexe

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Type complexe"
name	string	Local name of the type (empty if anonymous)
target namespace	string	Namespace URI of the type (empty if anonymous)
base type definition	Complex Type Definition	
final	Sequence of strings ("restriction"|"extension")	
context	Empty sequence (not implemented)	
derivation method	string ("restriction"|"extension")	
abstract	boolean	
attribute uses	Sequence of Attribute Use	
attribute wildcard	Optional Attribute Wildcard	
content type	function with properties: ("class":"Content Type", "variety":string ("element-only"|"empty"|"mixed"|"simple"), particle: optional Particle, "open content": function with properties ("class":"Open Content", "mode": string ("interleave"|"suffix"), "wildcard": Wildcard), "simple type definition": Type simple)	
prohibited substitutions	Sequence of strings ("restriction"|"extension")	
assertions	Sequence of Assertion	

Déclaration d'élément

Nom de propriété	Type de propriété	Valeur de propriété

kind	string	"Type complexe"
name	string	Local name of the type (empty if anonymous)
target namespace	string	Namespace URI of the type (empty if anonymous)
type definition	Type simple or Type complexe	
type table	function with properties ("class":"Type Table", "alternatives": sequence of Type Alternative, "default type definition": Type simple or Type complexe)	
scope	function with properties ("class":"Scope", "variety": ("global"|"local"), "parent": optional Type complexe)	
value constraint	see Attribute Declaration	
nillable	boolean	
identity-constraint definitions	Sequence of Identity Constraint	
substitution group affiliations	Sequence of Element Declaration	
substitution group exclusions	Sequence of strings ("restriction"|"extension")	
disallowed substitutions	Sequence of strings ("restriction"|"extension"|"substitution")	
abstract	boolean	

Caractère générique d'élément

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Wildcard"
namespace constraint	function with properties ("class": "Namespace Constraint", "variety": "any"|"enumeration"|"not", "namespaces": sequence of xs:anyURI, "disallowed names": list containing QNames and/or the strings "defined" and "definedSiblings"	
process contents	string ("strict"|"lax"|"skip")	

Facette

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	The name of the facet, for example "minLength" or

		"enumeration"
value	depends on facet	The value of the facet
fixed	boolean	
typed-value	For the enumeration facet only, array(xs:anyAtomicType*)	An array containing the enumeration values, each of which may in general be a sequence of atomic values. (Note: for the enumeration facet, the "value" property is a sequence of strings, regardless of the actual type)

Contrainte d'identité

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Identity-Constraint Definition"
name	string	Local name of the constraint
target namespace	string	Namespace URI of the constraint
identity-constraint category	string ("key"|"unique"|"keyRef")	
selector	XPath Property Record	
fields	Sequence of XPath Property Record	
referenced key	(For keyRef only): Identity Constraint	The corresponding key constraint

Groupe de modèle

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Groupe de modèle"
compositor	string ("sequence"|"choice"|"all")	
particles	Séquence de particule	

Définition de groupe de modèle

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Définition de groupe de modèle"
name	string	Nom local du groupe de modèle
target namespace	string	URI d'espace du groupe de modèle
model group	Groupe de modèle	

Notation

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Déclaration de notation"

name	string	Nom local de la notation
target namespace	string	URI d'espace de nom de la notation
system identifier	anyURI	
public identifier	string	

- Particule

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Particule"
min occurs	entier	
max occurs	entier ou string("unbounded")	
term	Déclaration d'élément, Caractère générique d'élément ou ModelGroup	

- Type simple

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Définition de type simple"
name	string	Nom local du type (vide si anonyme)
target namespace	string	URI d'espace de noms du type (vide si anonyme)
final	Séquence de string("restriction"|"extension"|"list"|"union")	
context	composant contenant	
base type definition	Type simple	
facets	Séquence de Facette	
fundamental facets	Séquence vide (pas implémentée)	
variety	string ("atomic"|"list"|"union")	
primitive type definition	Type simple	
item type definition	uniquement pour les types de liste) Type simple	
member type definitions	(uniquement pour les types d'union) Séquence de Type simple	

- Alternative de type

Nom de propriété	Type de propriété	Valeur de propriété

kind	string	"Type Alternative"
test	XPath Property Record	
type definition	Type simple ou Type complexe	

XPath Property Record

Nom de propriété	Type de propriété	Valeur de propriété
namespace bindings	Séquence des fonctions avec les propriétés ("prefix": string, "namespace": anyURI)	
default namespace	anyURI	
base URI	anyURI	L'URI de base statique de l'expression XPath
expression	string	L'expression XPath en tant que string

Schéma (deux arguments)

altova:schema(ComponentKind *as xs:string*, **Name** *as xs:QName*) **as (function(xs:string) as item()*)?** XP3.1 XQ3.1

Retourne le type du composant qui est spécifié dans le premier argument qui a un nom identique à celui fourni dans le second argument. Vous pouvez donc aller plus loin en sélectionnant une des propriétés du composant.

- Si cette propriété est un composant, vous pouvez aller encore plus en profondeur en sélectionnant une des propriétés de ce composant. Vous pouvez renouveler cette étape en allant plus loin dans le schéma.
- Si le composant est une valeur atomique, celle-ci sera retournée et vous ne pourrez pas aller plus loin.

Note: Dans des expressions XQuery, le schéma doit être importé explicitement. Dans des expressions XPath, le schéma doit avoir été importé dans l'environnement de traitement, par exemple dans XSLT avec l'instruction **xslt:import**.

Exemples

- ```
  import schema "" at "C:\Test\ExpReport.xsd";
  altova:schema("element declaration", xs:QName("OrgChart"))("type definition")
  ("content type")("particles")[3]!.("term")("kind")
  ```
  retourne la propriété `kind` du terme du troisième composant de `particles`. Ce composant de particules est un descendant de la déclaration d'élément ayant un `QName` de `OrgChart`

- ```
  import schema "" at "C:\Test\ExpReport.xsd";
  let $typedef := altova:schema("type definition", xs:QName("emailType"))
  for $facet in $typedef ("facets")
  return [$facet ("kind"), $facet("value")]
  ```

retourne, pour chaque **`facet`** de chaque composant **`emailType`**, un array contenant le type et la valeur de cette facette

Composants et leurs propriétés

- Assertion

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Assertion"
test	XPath Property Record	

- Déclaration d'attribut

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Attribute Declaration"
name	string	Local name of the attribute
target namespace	string	Namespace URI of the attribute
type definition	Type simple or Type complexe	
scope	A function with properties ("class":"Scope", "variety": "global" or "local", "parent": the containing Type complexe or Attribute Group)	
value constraint	If present, a function with properties ("class": "Value Constraint", "variety": "fixed" or "default", "value": atomic value, "lexical form": string. Note that the "value" property is not available for namespace-sensitive types	
inheritable	boolean	

- Déclaration de groupe d'attribut

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Attribute Group Definition"
name	string	Local name of the attribute group
target namespace	string	Namespace URI of the attribute group
attribute uses	Sequence of (Attribute Use)	
attribute wildcard	Optional Attribute Wildcard	

- Utilisation d'attribut

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Attribute Use"

required	boolean	true if the attribute is required, false if optional
value constraint	See Attribute Declaration	
inheritable	boolean	

- Caractère générique

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Wildcard"
namespace constraint	function with properties ("class": "Namespace Constraint", "variety": "any"|"enumeration"|"not", "namespaces": sequence of xs:anyURI, "disallowed names": list containing QNames and/or the strings "defined" and "definedSiblings"	
process contents	string ("strict"|"lax"|"skip")	

- Type complexe

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Type complexe"
name	string	Local name of the type (empty if anonymous)
target namespace	string	Namespace URI of the type (empty if anonymous)
base type definition	Complex Type Definition	
final	Sequence of strings ("restriction"|"extension")	
context	Empty sequence (not implemented)	
derivation method	string ("restriction"|"extension")	
abstract	boolean	
attribute uses	Sequence of Attribute Use	
attribute wildcard	Optional Attribute Wildcard	
content type	function with properties: ("class":"Content Type", "variety":string ("element-only"|"empty"|"mixed"|"simple"), particle: optional Particle, "open content": function with properties ("class":"Open Content", "mode": string ("interleave"|"suffix"), "wildcard": Wildcard), "simple type definition": Type	

	simple)	
prohibited substitutions	Sequence of strings ("restriction"|"extension")	
assertions	Sequence of Assertion	

- Déclaration d'élément

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Type complexe"
name	string	Local name of the type (empty if anonymous)
target namespace	string	Namespace URI of the type (empty if anonymous)
type definition	Type simple or Type complexe	
type table	function with properties ("class":"Type Table", "alternatives": sequence of Type Alternative, "default type definition": Type simple or Type complexe)	
scope	function with properties ("class":"Scope", "variety": ("global"|"local"), "parent": optional Type complexe)	
value constraint	see Attribute Declaration	
nillable	boolean	
identity-constraint definitions	Sequence of Identity Constraint	
substitution group affiliations	Sequence of Element Declaration	
substitution group exclusions	Sequence of strings ("restriction"|"extension")	
disallowed substitutions	Sequence of strings ("restriction"|"extension"|"substitution")	
abstract	boolean	

- Caractère générique d'élément

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Wildcard"
namespace constraint	function with properties ("class": "Namespace Constraint", "variety": "any"|"enumeration"|"not", "namespaces": sequence of xs:anyURI, "disallowed names": list containing QNames and/or the strings "defined"	

	and "definedSiblings"	
process contents	string ("strict"|"lax"|"skip")	

Facette

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	The name of the facet, for example "minLength" or "enumeration"
value	depends on facet	The value of the facet
fixed	boolean	
typed-value	For the enumeration facet only, array(xs:anyAtomicType*)	An array containing the enumeration values, each of which may in general be a sequence of atomic values. (Note: for the enumeration facet, the "value" property is a sequence of strings, regardless of the actual type)

Contrainte d'identité

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Identity-Constraint Definition"
name	string	Local name of the constraint
target namespace	string	Namespace URI of the constraint
identity-constraint category	string ("key"|"unique"|"keyRef")	
selector	XPath Property Record	
fields	Sequence of XPath Property Record	
referenced key	(For keyRef only): Identity Constraint	The corresponding key constraint

Groupe de modèle

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Groupe de modèle"
compositor	string ("sequence"|"choice"|"all")	
particles	Séquence de particule	

Définition de groupe de modèle

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Définition de groupe de modèle"

name	string	Nom local du groupe de modèle
target namespace	string	URI d'espace du groupe de modèle
model group	Groupe de modèle	

Notation

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Déclaration de notation"
name	string	Nom local de la notation
target namespace	string	URI d'espace de nom de la notation
system identifier	anyURI	
public identifier	string	

Particule

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Particule"
min occurs	entier	
max occurs	entier ou string("unbounded")	
term	Déclaration d'élément, Caractère générique d'élément ou ModelGroup	

Type simple

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Définition de type simple"
name	string	Nom local du type (vide si anonyme)
target namespace	string	URI d'espace de noms du type (vide si anonyme)
final	Séquence de string("restriction"|"extension"|"list"|"union")	
context	composant contenant	
base type definition	Type simple	
facets	Séquence de Facette	
fundamental facets	Séquence vide (pas implémentée)	
variety	string ("atomic"|"list"|"union")	
primitive type definition	Type simple	

item type definition	uniquement pour les types de liste) Type simple	
member type definitions	(uniquement pour les types d'union) Séquence de Type simple	

Alternative de type

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Type Alternative"
test	XPath Property Record	
type definition	Type simple ou Type complexe	

XPath Property Record

Nom de propriété	Type de propriété	Valeur de propriété
namespace bindings	Séquence des fonctions avec les propriétés ("prefix": string, "namespace": anyURI)	
default namespace	anyURI	
base URI	anyURI	L'URI de base statique de l'expression XPath
expression	string	L'expression XPath en tant que string

Type

`altova:type(Node` *`as item?`***`) as (function(xs:string) as item()*)?`** XP3.1 XQ3.1
La fonction **`altova:type`** soumet un nœud d'élément ou d'attribut d'un document XML et document et retourne l'information du type du nœud depuis le PSVI.

Note: Le document XML doit avoir une déclaration de schéma afin que ce schéma puisse être référencé.

Exemples

- ```
  for $element in //Email
  let $type := altova:type($element)
  return $type
  ```
  retourne une fonction qui contient l'information du type du nœud

- ```
  for $element in //Email
  let $type := altova:type($element)
  return $type ("kind")
  ```
  prend le composant du type du nœud (Type simple ou Type complexe) et retourne la valeur de la propriété **`kind`** du composant

Composants et leurs propriétés

- Assertion

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Assertion"
test	XPath Property Record	

- Déclaration d'attribut

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Attribute Declaration"
name	string	Local name of the attribute
target namespace	string	Namespace URI of the attribute
type definition	Type simple or Type complexe	
scope	A function with properties ("class":"Scope", "variety": "global" or "local", "parent": the containing Type complexe or Attribute Group)	
value constraint	If present, a function with properties ("class": "Value Constraint", "variety": "fixed" or "default", "value": atomic value, "lexical form": string. Note that the "value" property is not available for namespace-sensitive types	
inheritable	boolean	

- Déclaration de groupe d'attribut

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Attribute Group Definition"
name	string	Local name of the attribute group
target namespace	string	Namespace URI of the attribute group
attribute uses	Sequence of (Attribute Use)	
attribute wildcard	Optional Attribute Wildcard	

- Utilisation d'attribut

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Attribute Use"
required	boolean	true if the attribute is required, false if optional

value constraint	See Attribute Declaration	
inheritable	boolean	

Caractère générique

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Wildcard"
namespace constraint	function with properties ("class": "Namespace Constraint", "variety": "any"|"enumeration"|"not", "namespaces": sequence of xs:anyURI, "disallowed names": list containing QNames and/or the strings "defined" and "definedSiblings"	
process contents	string ("strict"|"lax"|"skip")	

Type complexe

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Type complexe"
name	string	Local name of the type (empty if anonymous)
target namespace	string	Namespace URI of the type (empty if anonymous)
base type definition	Complex Type Definition	
final	Sequence of strings ("restriction"|"extension")	
context	Empty sequence (not implemented)	
derivation method	string ("restriction"|"extension")	
abstract	boolean	
attribute uses	Sequence of Attribute Use	
attribute wildcard	Optional Attribute Wildcard	
content type	function with properties: ("class":"Content Type", "variety":string ("element-only"|"empty"|"mixed"|"simple"), particle: optional Particle, "open content": function with properties ("class":"Open Content", "mode": string ("interleave"|"suffix"), "wildcard": Wildcard), "simple type definition": Type simple)	
prohibited	Sequence of strings	

substitutions	("restriction"|"extension")	
assertions	Sequence of Assertion	

Déclaration d'élément

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Type complexe"
name	string	Local name of the type (empty if anonymous)
target namespace	string	Namespace URI of the type (empty if anonymous)
type definition	Type simple or Type complexe	
type table	function with properties ("class":"Type Table", "alternatives": sequence of Type Alternative, "default type definition": Type simple or Type complexe)	
scope	function with properties ("class":"Scope", "variety": ("global"|"local"), "parent": optional Type complexe)	
value constraint	see Attribute Declaration	
nillable	boolean	
identity-constraint definitions	Sequence of Identity Constraint	
substitution group affiliations	Sequence of Element Declaration	
substitution group exclusions	Sequence of strings ("restriction"|"extension")	
disallowed substitutions	Sequence of strings ("restriction"|"extension"|"substitution")	
abstract	boolean	

Caractère générique d'élément

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Wildcard"
namespace constraint	function with properties ("class": "Namespace Constraint", "variety": "any"|"enumeration"|"not", "namespaces": sequence of xs:anyURI, "disallowed names": list containing QNames and/or the strings "defined" and "definedSiblings"	
process contents	string ("strict"|"lax"|"skip")	

- Facette

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	The name of the facet, for example "minLength" or "enumeration"
value	depends on facet	The value of the facet
fixed	boolean	
typed-value	For the enumeration facet only, array(xs:anyAtomicType*)	An array containing the enumeration values, each of which may in general be a sequence of atomic values. (Note: for the enumeration facet, the "value" property is a sequence of strings, regardless of the actual type)

- Contrainte d'identité

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Identity-Constraint Definition"
name	string	Local name of the constraint
target namespace	string	Namespace URI of the constraint
identity-constraint category	string ("key"|"unique"|"keyRef")	
selector	XPath Property Record	
fields	Sequence of XPath Property Record	
referenced key	(For keyRef only): Identity Constraint	The corresponding key constraint

- Groupe de modèle

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Groupe de modèle"
compositor	string ("sequence"|"choice"|"all")	
particles	Séquence de particule	

- Définition de groupe de modèle

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Définition de groupe de modèle"
name	string	Nom local du groupe de modèle
target namespace	string	URI d'espace du groupe de modèle

model group	Groupe de modèle	

Notation

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Déclaration de notation"
name	string	Nom local de la notation
target namespace	string	URI d'espace de nom de la notation
system identifier	anyURI	
public identifier	string	

Particule

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Particule"
min occurs	entier	
max occurs	entier ou string("unbounded")	
term	Déclaration d'élément, Caractère générique d'élément ou ModelGroup	

Type simple

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Définition de type simple"
name	string	Nom local du type (vide si anonyme)
target namespace	string	URI d'espace de noms du type (vide si anonyme)
final	Séquence de string("restriction"|"extension"|"list"|"union")	
context	composant contenant	
base type definition	Type simple	
facets	Séquence de Facette	
fundamental facets	Séquence vide (pas implémentée)	
variety	string ("atomic"|"list"|"union")	
primitive type definition	Type simple	
item type definition	uniquement pour les types de liste) Type simple	

member type definitions	(uniquement pour les types d'union) Séquence de Type simple	

- Alternative de type

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Type Alternative"
test	XPath Property Record	
type definition	Type simple ou Type complexe	

- XPath Property Record

Nom de propriété	Type de propriété	Valeur de propriété
namespace bindings	Séquence des fonctions avec les propriétés ("prefix": string, "namespace": anyURI)	
default namespace	anyURI	
base URI	anyURI	L'URI de base statique de l'expression XPath
expression	string	L'expression XPath en tant que string

12.2.2.1.7 Fonctions XPath/XQuery : Séquence

Les fonctions d'extension de la séquence d'Altova peuvent être utilisées dans les expressions XPath et XQuery et proposent des fonctions supplémentaires pour le traitement des données. Les fonctions dans cette section peuvent être utilisées avec les moteurs **XPath 3.0** et **XQuery 3.0** d'Altova. Ils sont disponibles dans des contextes XPath/XQuery.

Note concernant le nommage de fonctions et de l'applicabilité de la langue

Les fonctions d'extension Altova peuvent être utilisées dans les expressions XPath/XQuery. Elles fournissent des fonctions supplémentaires aux fonctions d'ores et déjà disponibles dans la librairie standard des fonctions XPath, XQuery et XSLT. Les fonctions d'extension Altova se trouvent dans **l'espace de nom des fonctions d'extension Altova, `http://www.altova.com/xslt-extensions`**, et sont indiquées dans cette section par le préfixe **`altova:`**, qui est présupposé être lié à cet espace de nom. Veuillez noter que, en ce qui concerne les versions futures de votre produit, la prise en charge d'une fonction peut être interrompue et le comportement de certaines fonctions peut changer. Veuillez consulter la documentation lors des publications à venir pour plus d'informations concernant la prise en charge des fonctions d'extension Altova de cette version.

Fonctions XPath (utilisées dans les expressions XPath dans XSLT) :	XP1 XP2 XP3.1
Fonctions XSLT (utilisées dans les expressions XPath dans XSLT) :	XSLT1 XSLT2 XSLT3
Fonctions XQuery (utilisées dans les expressions XQuery dans XQuery) :	XQ1 XQ3.1

▼ attributes [altova:]

altova:attributes(AttributeName *as xs:string***) asattribute()*** XP3.1 XQ3.1
Retourne tous les attributs possédant un nom local qui est le même que le nom fourni dans l'argument d'entrée, `AttributeName`. La recherche est sensible à la casse et est conduite le long de l'axe `attribute::` . Cela signifie que le nœud contextuel doit être le nœud d'élément parent.

Exemples

- **altova:attributes**("MyAttribute") retourne `MyAttribute()*`

altova:attributes(AttributeName *as xs:string*, **SearchOptions** *as xs:string***) asattribute()*** XP3.1 XQ3.1
Retourne tous les attribut possédant un nom local qui est le même que le nom fourni dans l'argument d'entrée, `AttributeName`. La recherche est sensible à la casse et est conduite le long de l'axe `attribute::` . Le nœud contextuel doit être le nœud d'élément parent. Le deuxième argument est une chaîne contenant des flags optionnels. Les flags disponibles sont :
r = passe à une recherche d'expression régulière ; `AttributeName` doit alors être une chaîne de recherche d'expression régulière ;
f = si cette option est spécifiée, alors `AttributeName` fournit une concordance complète ; dans le cas contraire, `AttributeName` ne nécessite qu'une concordance partielle d'un nom d'attribut pour retourner cet attribut. Par exemple : si **f** n'est pas spécifié, `MyAtt` retournera `MyAttribute`;
i = passe à une recherche insensible à la casse ;
p = comprend le préfixe d'espace de nom dans la recherche ; `AttributeName` devrait ensuite contenir le préfixe d'espace de nom, par exemple : `altova:MyAttribute`.
Les flags peuvent être écrits dans n'importe quel ordre. Les flags invalides généreront des erreurs. Un ou plusieurs flags peuvent être omis. La chaîne vide est permise et produire le même effet que la fonction n'ayant qu'un seul argument (*signature précédente*). Néanmoins, une séquence vide n'est pas permise en tant que le deuxième argument.

Exemples

- **altova:attributes**("MyAttribute", "rfip") retourne `MyAttribute()*`
- **altova:attributes**("MyAttribute", "pri") retourne `MyAttribute()*`
- **altova:attributes**("MyAtt", "rip") retourne `MyAttribute()*`
- **altova:attributes**("MyAttributes", "rfip") ne retourne aucune correspondance.
- **altova:attributes**("MyAttribute", "") retourne `MyAttribute()*`
- **altova:attributes**("MyAttribute", "Rip") retourne une erreur de flag non reconnu.
- **altova:attributes**("MyAttribute",) retourne une erreur d'argument manquant.

▼ elements [altova:]

altova:elements(ElementName *as xs:string***) aselement()*** XP3.1 XQ3.1
Retourne tous les éléments qui ont un nom local identique au nom fourni dans l'argument d'entrée,

`ElementName`. La recherche est sensible à la casse et est conduite le long de l'axe `child::`. Le nœud contextuel doit être le nœud parent de/s l'élément/s recherché.

Exemples

- **`altova:elements`**`("MyElement")` retourne `MyElement()*`

`altova:elements(ElementName` *`as xs:string`*`, SearchOptions` *`as xs:string`*`) aselement()*` XP3.1 XQ3.1

Retourne tous les éléments qui ont un nom local identique au nom fourni dans l'argument d'entrée, `ElementName`. La recherche est sensible à la casse et est conduite le long de l'axe `child::`. Le nœud contextuel doit être le nœud parent de/s l'élément/s recherché. Le second argument est une chaîne contenant des flags optionnels. Les flags disponibles sont :
`r` = passe à une recherche d'expression régulière ; `ElementName` doit alors être une chaîne de recherche d'expression régulière ;
`f` = si cette option est spécifiée, alors `ElementName` fournit une concordance complète ; dans le cas contraire, `ElementName` ne nécessite qu'une concordance partielle d'un nom d'élément pour retourner cet élément. Par exemple : si **`f`** n'est pas spécifié, `MyElem` retournera `MyElement` ;
`i` = passe à une recherche insensible à la casse ;
`p` = comprend le préfixe d'espace de nom dans la recherche ; `ElementName` devrait ensuite contenir le préfixe d'espace de nom, par exemple : `altova:MyElement`.
Les flags peuvent être écrits dans n'importe quel ordre. Les flags invalides généreront des erreurs. Un ou plusieurs flags peuvent être omis. La chaîne vide est autorisée et produira le même effet que la fonction n'ayant qu'un argument (*signature précédente*). Néanmoins, une séquence vide n'est pas autorisée.

Exemples

- **`altova:elements`**`("MyElement", "rip")` retourne `MyElement()*`
- **`altova:elements`**`("MyElement", "pri")` retourne `MyElement()*`
- **`altova:elements`**`("MyElement", "")` retourne `MyElement()*`
- **`altova:elements`**`("MyElem", "rip")` retourne `MyElement()*`
- **`altova:elements`**`("MyElements", "rfip")` retourne aucune correspondance
- **`altova:elements`**`("MyElement", "Rip")` retourne une erreur flag-non reconnu.
- **`altova:elements`**`("MyElement", )` retourne une erreur second-argument-manquant.

find-first [altova:]

`altova:find-first((Sequence` *`as item()*`**`), (Condition( Sequence-Item`** *`as xs:boolean`*`))`**
`asitem()?` XP3.1 XQ3.1

Cette fonction prend deux arguments. Le premier argument est une séquence d'un ou de plusieurs items de tout type de données. Le second argument, `Condition`, est une référence à une fonction XPath qui prend un argument (possède une arité de **`1`**) et retourne un booléen. Chaque item de **`Sequence`** est soumis à son tour à la fonction référencée dans `Condition`. (*Rappel :* cette fonction prend un seul argument.) Le premier item **`Sequence`** qui cause la fonction dans **`Condition`** à évaluer à **`true()`** est retourné en tant que le résultat de **`altova:find-first`**, et l'itération s'arrête.

Exemples

- **`altova:find-first`**`(5 to 10, function($a) {$a mod 2 = 0})` retourne `xs:integer 6`
  L'argument **`Condition`** référence la fonction en ligne XPath 3.0, **`function()`**, qui déclare une fonction en ligne nommée **`$a`** puis la définit. Chaque item dans l'argument **`Sequence`** de **`altova:find-first`** est passé à son tour à **`$a`** en tant que sa valeur d'entrée. La valeur d'entrée est testée sur la condition dans la définition de la fonction (`$a mod 2 = 0`). La première valeur d'entrée pour satisfaire cette

condition est retournée en tant que le résultat de **altova:find-first** (dans ce cas **6**).

- **altova:find-first**((1 to 10), (function($a) {$a+3=7})) retourne xs:integer 4

Autres exemples
Si le fichier **C:\Temp\Customers.xml** existe :
- **altova:find-first**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) retourne xs:string C:\Temp\Customers.xml

Si le fichier **C:\Temp\Customers.xml** n'existe pas et que **http://www.altova.com/index.html** existe :
- **altova:find-first**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) retourne xs:string http://www.altova.com/index.html

Si le fichier **C:\Temp\Customers.xml** n'existe pas, et que **http://www.altova.com/index.html** n'existe pas non plus :
- **altova:find-first**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) ne retourne aucun résultat

Notes à propos des exemples indiqués ci-dessus

- La fonction XPath 3.0, doc-available, prend un seul argument de chaîne, qui est utilisé en tant qu'URI, et retourne true si un nœud de document est trouvé à l'URI soumis. (Le document à l'URI soumis doit donc être un document XML.)
- La fonction doc-available peut être utilisée pour **Condition**, le second argument de altova:find-first, parce qu'il ne prend qu'un seul argument (arité=1), parce qu'il prend un item() en tant qu'entrée (une chaîne qui est utilisée en tant qu'URI), et retourne une valeur booléenne.
- Veuillez noter que la fonction doc-available est uniquement référencée, elle n'est pas appelée. Le suffixe #1 qui y est attaché indique une fonction avec une arité de 1. Sous sa forme complète, doc-available#1 signifie simplement : *Utiliser la fonction doc-availabe() à l'arité=1, en l'y passant en tant que son seul argument, chacun à son tour, chacun des items dans la première séquence*. En résultat, chacun des deux chaînes sera passée à **doc-available()**, qui utilise la chaîne en tant qu'URI et teste si un nœud de document existe à l'URI. S'il en existe un, **doc-available()** évalue à true() et cette chaîne est retournée en tant que le résultat de la fonction **altova:find-first**. *Note à propos de la fonction doc-available() : les chemins relatifs sont résolus relativement à l'URI de base actuel, qui est par défaut l'URI du document XML à partir duquel la fonction est chargée.*

find-first-combination [altova:]

altova:find-first-combination((Seq-01 *as item()**), **(Seq-02** *as item()**), **(Condition(Seq-01-Item, Seq-02-Item** *as xs:boolean*)) **asitem()*** XP3.1 XQ3.1

Cette fonction prend trois arguments :

- Les deux premiers arguments, **Seq-01** et **Seq-02**, sont des séquences d'un ou de plusieurs items de tout type de données.
- Le troisième argument, **Condition**, est une référence à une fonction XPath qui prend deux arguments (a une arité de **2**) et retourne un booléen.

Les items de **Seq-01** et **Seq-02** sont passés dans des paires ordonnées (un item de chaque séquence faisant une paire) en tant que les arguments de la fonction dans **Condition**. Les paires sont classées comme suit :

```
If   Seq-01 = X1, X2, X3 ... Xn
And  Seq-02 = Y1, Y2, Y3 ... Yn
Then (X1 Y1), (X1 Y2), (X1 Y3) ... (X1 Yn), (X2 Y1), (X2 Y2) ... (Xn Yn)
```

La première paire ordonnée qui entraîne la fonction **Condition** à évaluer à **true()** est retournée en tant que le résultat de **altova:find-first-combination**. Veuillez noter que : (i) si la fonction **Condition** itère par le biais des paires d'argument soumises et n'évalue pas une fois à **true()**, alors **altova:find-first-combination** retournera *Aucun résultat* ; (ii) Le résultat de **altova:find-first-combination** sera toujours une paire d'items (de tout type de données) ou aucun item.

Exemples

- **altova:find-first-combination**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 32}) retourne la séquence de xs:integers (11, 21)
- **altova:find-first-combination**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 33}) retourne la séquence de f xs:integers (11, 22)
- **altova:find-first-combination**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 34}) retourne la séquence de xs:integers (11, 23)

find-first-pair [altova:]

altova:find-first-pair((Seq-01 *as item()**), **(Seq-02** *as item()**), **(Condition(Seq-01-Item, Seq-02-Item** *as xs:boolean*)) **asitem()*** XP3.1 XQ3.1

Cette fonction prend trois arguments :

- Les deux premiers arguments, **Seq-01** et **Seq-02**, sont des séquences d'un ou de plusieurs items de tout type de données.
- Le troisième argument, **Condition**, est une référence à une fonction XPath qui prend deux arguments (a une arité de **2**) et retourne un booléen.

Les items de **Seq-01** et **Seq-02** sont passés dans des paires ordonnées en tant que les arguments de la fonction dans **Condition**. Les paires sont classées comme suit :

```
If   Seq-01 = X1, X2, X3 ... Xn
And  Seq-02 = Y1, Y2, Y3 ... Yn
Then (X1 Y1), (X2 Y2), (X3 Y3) ... (Xn Yn)
```

La première paire ordonnée qui cause la fonction **Condition** à évaluer à **true()** est retournée en tant que le résultat de **altova:find-first-pair**. Veuillez noter que : (i) Si la fonction **Condition** itère par le biais des paires d'arguments soumis et n'évalue pas une seule fois à **true()**, alors **altova:find-first-pair** retournera *Aucun résultat*; (ii) Le résultat de **altova:find-first-pair** sera toujours une paire d'items (de tout type de données) ou aucun item.

Exemples

- **altova:find-first-pair**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 32}) retourne la séquence de xs:integers (11, 21)
- **altova:find-first-pair**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 33}) retourne *Aucun résultat*

Veuillez noter à partir des deux exemples ci-dessus que l'ordonnance des paires est : `(11, 21) (12, 22) (13, 23)...(20, 30)`. C'est pourquoi le second exemple retourne *Aucun résultat* (parce qu'aucune paire ordonnée de donne une somme de 33).

find-first-pair-pos [altova:]

altova:find-first-pair-pos((Seq-01 *as item()**), **(Seq-02** *as item()**), **(Condition(Seq-01-Item, Seq-02-Item** *as xs:boolean*)) **asxs:integer** XP3.1 XQ3.1

Cette fonction prend trois arguments :

- Les deux premiers arguments, **Seq-01** and **Seq-02**, sont des séquences d'un ou de plusieurs items de tout type de données.
- Le troisième argument, **Condition**, est une référence à une fonction XPath qui prend deux arguments (a une arité de **2**) et retourne un booléen.

Les items de **Seq-01** et **Seq-02** sont passés dans des paires ordonnées en tant que les arguments de la fonction dans **Condition**. Les paires sont classées comme suit :

```
If   Seq-01 = X1, X2, X3 ... Xn
And  Seq-02 = Y1, Y2, Y3 ... Yn
Then (X1 Y1), (X2 Y2), (X3 Y3) ... (Xn Yn)
```

La position d'index de la première paire ordonnée qui entraîne la fonction **Condition** à évaluer à **true()** est retournée en tant que le résultat de **altova:find-first-pair-pos**. Veuillez noter que si la fonction **Condition** itère par le biais des paires d'arguments soumises et n'évalue pas une seule fois à **true()**, alors **altova:find-first-pair-pos** retournera *Aucun résultat*.

Exemples

- `altova:find-first-pair-pos(11 to 20, 21 to 30, function($a, $b) {$a+$b = 32})` retourne `1`
- `altova:find-first-pair-pos(11 to 20, 21 to 30, function($a, $b) {$a+$b = 33})` retourne *Aucun résultat*

Veuillez noter à partir des deux exemples ci-dessus que l'ordonnance des paires est : `(11, 21) (12, 22) (13, 23)...(20, 30)`. dans le premier exemple, la première paire entraîne la fonction **Condition** à évaluer à **true()**, et donc sa position d'index dans la séquence, **1**, est retournée. Le second exemple retourne *Aucun résultat* parce qu'aucune paire ne totalise pas une somme de 33.

find-first-pos [altova:]

altova:find-first-pos((Sequence *as item()**), **(Condition(Sequence-Item** *as xs:boolean*)) **asxs:integer** XP3.1 XQ3.1

Cette fonction prend deux arguments. Le premier argument est une séquence d'un ou de plusieurs items de tout type de données. Le second argument, `Condition`, est une référence à une fonction XPath qui prend un argument (a une arité de **1**) et retourne une booléenne. Chaque item de **Sequence** est soumis à son tour à la fonction référencée dans `Condition`. (*Rappel :* cette fonction prend un seul argument.) Le premier item **Sequence** qui cause la fonction dans **Condition** à évaluer à **true()** voit sa position index dans **Sequence** retournée en tant que résultat de **altova:find-first-pos**, et l'itération stoppe.

Exemples

- **altova:find-first-pos**(5 to 10, function($a) {$a mod 2 = 0}) retourne xs:integer 2

L'argument **Condition** référence la fonction en ligne XPath 3.0, **function()**, qui déclare une fonction en ligne nommée **$a** et puis la définit. Chaque item dans l'argument de **Sequence** de **altova:find-first-pos** est passé à son tour à **$a** en tant que sa valeur d'entrée. La valeur d'entrée est testée à la condition dans la définition de la fonction ($a mod 2 = 0). La position d'index dans la séquence de la première valeur d'entrée pour satisfaire à cette condition est retournée en tant que le résultat de **altova:find-first-pos** (dans ce cas **2**, puisque **6**, la première valeur (dans la séquence) afin de satisfaire à la condition, est à la position d'index **2** dans la séquence).

- **altova:find-first-pos**((2 to 10), (function($a) {$a+3=7})) retourne xs:integer 3

Autres exemples

Si le fichier **C:\Temp\Customers.xml** existe :

- **altova:find-first-pos**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) retourne 1

Si le fichier **C:\Temp\Customers.xml** n'existe pas, et que **http://www.altova.com/index.html** existe :

- **altova:find-first-pos**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) retourne 2

Si le fichier **C:\Temp\Customers.xml** n'existe pas, et que **http://www.altova.com/index.html** n'existe pas non plus :

- **altova:find-first-pos**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) ne retourne aucun résultat

Notes à propos des exemples donnés ci-dessus

- La fonction XPath 3.0, doc-available, prend un seul argument de chaîne qui est utilisé en tant qu'un URI, et retourne true si un nœud de document est trouvé à l'URI soumis. (Le document à l'URI soumis doit donc être un document XML.)
- La fonction doc-available peut être utilisée pour **Condition**, le second argument de altova:find-first-pos, parce qu'il ne prend qu'un seul argument (arité=1), parce qu'il prend un item() en tant qu'entrée (une chaîne qui est utilisée en tant qu'un URI), et retourne une valeur booléenne.
- Veuillez noter que la fonction doc-available est uniquement référencée, elle n'est pas appelée. Le suffixe #1 qui y est attaché indique une fonction avec une arité de 1. dans sa totalité, doc-available#1 signifie simplement : *Utiliser la fonction doc-availabe() qui a arité=1, y passant, en tant que son argument simple, chacun à son tour, chaque item dans la première séquence*. En tant que résultat. chacune des deux chaînes sera passée à **doc-available()**, qui utilise la chaîne en tant qu'un URI et teste si un nœud de document existe à l'URI. Si c'est le cas, la fonction **doc-available()** évalue à true() et la position de l'index de cette chaîne dans la séquence est retournée en tant que le résultat de la fonction **altova:find-first-pos**. *Note à propos de la fonction doc-available() : les chemins relatifs sont résolus relativement à l'URI de base actuel, qui par défaut est l'URI du document XML à partir duquel la fonction est chargée.*

for-each-attribute-pair [altova:]

altova:for-each-attribute-pair(Seq1 *as element()?*, **Seq2** *as element()?*, **Function** *as function()*) **asitem()*** XP3.1 XQ3.1

Les premiers deux arguments identifient deux éléments, dont les attributs sont utilisés pour générer des paires d'attribut, dans laquelle un attribut d'une paire est obtenu depuis le premier élément et l'autre attribut est obtenu depuis le second élément. Les paires d'attribut sont sélectionnées sur le fait qu'ils présentent le même nom, et les paires sont classées par ordre alphabétique (sur leur nom) dans un ensemble. Si, pour un attribut, aucun attribut correspondant n'existe dans l'autre élément, la paire sera "disjointe", ce qui signifie qu'elle consiste en un seul membre. L'item de fonction (troisième argument `Function`) est appliqué séparément à chaque paire dans la séquence des paires (joints et disjointe), résultant en une sortie qui est une séquence d'items.

Exemples

- **altova:for-each-attribute-pair**(`/Example/Test-A, /Example/Test-B, function($a, $b) {$a+b}`) retourne ...

  (2, 4, 6) si
  <Test-A att1="1" att2="2" att3="3" />
  <Test-B att1="1" att2="2" att3="3" />

  (2, 4, 6) si
  <Test-A att2="2" att1="1" att3="3" />
  <Test-B att3="3" att2="2" att1="1" />

  (2, 6) si
  <Test-A att4="4" att1="1" att3="3" />
  <Test-B att3="3" att2="2" att1="1" />

  Note : *Le résultat* **(2 6)** *est obtenu par le biais de l'action suivante :* **(1+1 ()+2 3+3 4+())**. *Si un des opérandes est la séquence vide, comme c'est le cas des items 2 et 4, le résultat de l'addition est une séquence vide.*

- **altova:for-each-attribute-pair**(`/Example/Test-A, /Example/Test-B, concat#2`) retourne ...

  (11 22 33) si
  <Test-A att1="1" att2="2" att3="3" />
  <Test-B att1="1" att2="2" att3="3" />

  (11 2 33 4) si
  <Test-A att4="4" att1="1" att3="3" />
  <Test-B att3="3" att2="2" att1="1" />

for-each-combination [altova:]

altova:for-each-combination(FirstSequence *as item()**, **SecondSequence** *as item()**, **Function($i,$j){$i || $j}) asitem()*** XP3.1 XQ3.1

Les items des deux séquences dans les deux premiers arguments sont combinés de manière à ce que chaque item de la première séquence est combiné, dans l'ordre, une fois avec chaque item de la seconde séquence. La fonction donnée en tant que le troisième argument est appliquée à chaque combinaison dans la séquence résultante, entraînant une sortie qui est une séquence d'items (*voir exemple*).

Exemples

- **altova:for-each-combination**(`('a', 'b', 'c'), ('1', '2', '3'), function($i, $j)`

`{$i || $j}`) retourne `('a1', 'a2', 'a3', 'b1', 'b2', 'b3', 'c1', 'c2', 'c3')`

for-each-matching-attribute-pair [altova:]

altova:for-each-matching-attribute-pair(Seq1 *as element()?*, **Seq2** *as element()?*, **Function** *as function()*) **asitem()*** XP3.1 XQ3.1

Les premiers deux arguments identifient deux éléments, dont les attributs sont utilisés pour générer des paires d'attribut, dans laquelle un attribut d'une paire est obtenu depuis le premier élément et l'autre attribut est obtenu depuis le second élément. Les paires d'attribut sont sélectionnées sur le fait qu'ils présentent le même nom, et les paires sont classées par ordre alphabétique (sur leur nom) dans un ensemble. Si, pour un attribut, aucun attribut correspondant n'existe dans l'autre élément, aucune paire ne sera générée. L'item de fonction (troisième argument `Function`) est appliqué séparément à chaque paire dans la séquence des paires, résultant en une sortie qui est une séquence d'items.

Exemples

- **altova:for-each-matching-attribute-pair**(`/Example/Test-A, /Example/Test-B, function($a, $b){$a+b}`) retourne ...

  (2, 4, 6) si
  <Test-A att1="1" att2="2" att3="3" />
  <Test-B att1="1" att2="2" att3="3" />

  (2, 4, 6) si
  <Test-A att2="2" att1="1" att3="3" />
  <Test-B att3="3" att2="2" att1="1" />

  (2, 6) si
  <Test-A att4="4" att1="1" att3="3" />
  <Test-B att3="3" att2="2" att3="1" />

- **altova:for-each-matching-attribute-pair**(`/Example/Test-A, /Example/Test-B, concat#2`) retourne ...

  (11, 22, 33) si
  <Test-A att1="1" att2="2" att3="3" />
  <Test-B att1="1" att2="2" att3="3" />

  (11, 33) si
  <Test-A att4="4" att1="1" att3="3" />
  <Test-B att3="3" att2="2" att1="1" />

substitute-empty [altova:]

altova:substitute-empty(FirstSequence *as item()**, **SecondSequence** *as item()*) **asitem()*** XP3.1 XQ3.1

Si `FirstSequence` est vide, retourne `SecondSequence`. Si `FirstSequence` n'est pas vide, retourne `FirstSequence`.

Exemples

- **altova:substitute-empty**(`(1,2,3), (4,5,6)`) retourne `(1,2,3)`

- **altova:substitute-empty**((), (4,5,6)) retourne (4,5,6)

12.2.2.1.8 Fonctions XPath/XQuery : Chaîne

Les fonctions d'extension de chaîne d'Altova peuvent être utilisées dans les expressions XPath et XQuery et proposent des fonctions supplémentaires pour le traitement des données. Les fonctions dans cette section peuvent être utilisées avec les moteurs **XPath 3.0** et **XQuery 3.0** d'Altova. Ils sont disponibles dans des contextes XPath/XQuery.

Note concernant le nommage de fonctions et de l'applicabilité de la langue

Les fonctions d'extension Altova peuvent être utilisées dans les expressions XPath/XQuery. Elles fournissent des fonctions supplémentaires aux fonctions d'ores et déjà disponibles dans la librairie standard des fonctions XPath, XQuery et XSLT. Les fonctions d'extension Altova se trouvent dans **l'espace de nom des fonctions d'extension Altova, http://www.altova.com/xslt-extensions**, et sont indiquées dans cette section par le préfixe **altova:**, qui est présupposé être lié à cet espace de nom. Veuillez noter que, en ce qui concerne les versions futures de votre produit, la prise en charge d'une fonction peut être interrompue et le comportement de certaines fonctions peut changer. Veuillez consulter la documentation lors des publications à venir pour plus d'informations concernant la prise en charge des fonctions d'extension Altova de cette version.

Fonctions XPath (utilisées dans les expressions XPath dans XSLT) :	XP1 XP2 XP3.1
Fonctions XSLT (utilisées dans les expressions XPath dans XSLT) :	XSLT1 XSLT2 XSLT3
Fonctions XQuery (utilisées dans les expressions XQuery dans XQuery) :	XQ1 XQ3.1

▼ camel-case [altova:]

altova:camel-case(InputString *as xs:string***) asxs:string** XP3.1 XQ3.1
Retourne la chaîne d'entrée **InputString** en CamelCase. La chaîne est analysée en utilisant l'expression régulière `'\s'` (qui est un raccourci pour le caractère d'espace blanc). Le premier caractère non-espace blanc après un espace blanc ou une séquence de plusieurs espaces blancs est mis en majuscule. Le premier caractère dans la chaîne de sortie est mis en majuscule.

Exemples

- **altova:camel-case**("max") retourne Max
- **altova:camel-case**("max max") retourne Max Max
- **altova:camel-case**("file01.xml") retourne File01.xml
- **altova:camel-case**("file01.xml file02.xml") retourne File01.xml File02.xml
- **altova:camel-case**("file01.xml file02.xml") retourne File01.xml File02.xml
- **altova:camel-case**("file01.xml -file02.xml") retourne File01.xml -file02.xml

altova:camel-case(InputString *as xs:string*, **SplitChars** *as xs:string*, **IsRegex** *as*

xs:boolean) **asxs:string** XP3.1 XQ3.1
Convertit la chaîne d'entrée **InputString** en camel case en utilisant **SplitChars** pour déterminer le/s caractère/s qui déclenche/nt la prochaine mise en majuscule. **SplitChars** est utilisé en tant qu'expression régulière quand **IsRegex = true()**, ou en tant que caractères normaux quand **IsRegex = false()**. Le premier caractère dans la chaîne de sortie est mis en majuscule.

Exemples

- `altova:camel-case("setname getname", "set|get", true())` retourne `setName getName`
- `altova:camel-case("altova\documents\testcases", "\", false())` retourne `Altova\Documents\Testcases`

char [altova:]

altova:char(Position *as xs:integer*) **asxs:string** XP3.1 XQ3.1
Retourne une chaîne contenant le caractère à la position spécifiée par l'argument `Position`, dans la chaîne obtenue en convertissant la valeur de l'item contextuel en `xs:string`. La chaîne de résultat sera vide si aucun caractère n'existe à l'index soumis par l'argument `Position`.

Exemples

Si l'item contextuel est **1234ABCD**:

- `altova:char(2)` retourne `2`
- `altova:char(5)` retourne `A`
- `altova:char(9)` retourne la chaîne vide.
- `altova:char(-2)` retourne la chaîne vide.

altova:char(InputString *as xs:string*, **Position** *as xs:integer*) **asxs:string** XP3.1 XQ3.1
Retourne une chaîne contenant la caractère à la position spécifiée par l'argument `Position`, dans la chaîne soumise en tant que l'argument `InputString`. La chaîne de résultat sera vide si aucun caractère n'existe à l'index soumis par l'argument `Position`.

Exemples

- `altova:char("2014-01-15", 5)` retourne `-`
- `altova:char("USA", 1)` retourne `U`
- `altova:char("USA", 10)` retourne la chaîne vide.
- `altova:char("USA", -2)` retourne la chaîne vide.

create-hash-from-string[altova:]

altova:create-hash-from-string(InputString *as xs:string*) **asxs:string** XP2 XQ1 XP3.1 XQ3.1
altova:create-hash-from-string(InputString *as xs:string*, **HashAlgo** *as xs:string*) **asxs:string** XP2 XQ1 XP3.1 XQ3.1
Génère un string de hashage depuis `InputString` en utilisant l'algorithme de hashage spécifié par l'argument `HashAlgo`. Les algorithmes de hashage suivants peuvent être spécifiés (en majuscule ou en minuscule) : **MD5**, **SHA-1**, **SHA-224**, **SHA-256**, **SHA-384**, **SHA-512**. Si le deuxième argument n'est pas spécifié (*voir la première signature ci-dessus*), l'algorithme de hashage **SHA-256** sera utilisé.

Exemples

- `altova:create-hash-from-string('abc')` retourne un string de hashage généré en utilisant l'algorithme de hachage **SHA-256**.
- `altova:create-hash-from-string('abc', 'md5')` retourne un string de hashage généré en

utilisant l'algorithme de hachage **MD5**.
- **altova:create-hash-from-string**('abc', 'MD5') retourne un string de hashage généré en utilisant l'algorithme de hachage **MD5**.

▼ first-chars [altova:]

altova:first-chars(X-Number *as xs:integer*) **asxs:string** XP3.1 XQ3.1
Retourne une chaîne contenant le premier X-Number des caractères de la chaîne obtenue en convertissant la valeur de l'item de contexte en xs:string.

Exemples

Si l'item de contexte est **1234ABCD** :
- **altova:first-chars**(2) retourne 12
- **altova:first-chars**(5) retourne 1234A
- **altova:first-chars**(9) retourne 1234ABCD

altova:first-chars(InputString *as xs:string*, **X-Number** *as xs:integer*) **asxs:string** XP3.1 XQ3.1
Retourne une chaîne contenant le premier X-Number des caractères de la chaîne soumise en tant que l'argument InputString.

Exemples

- **altova:first-chars**("2014-01-15", 5) retourne 2014-
- **altova:first-chars**("USA", 1) retourne U

▼ format-string [altova:]

altova:format-string(InputString *as xs:string*, **FormatSequence** *as item()**) **asxs:string** XP3.1 XQ3.1
Le string d'entrée (premier argument) contient des paramètres de position (%1, %2, etc). Chaque paramètre est remplacé par l'item de string qui est situé dans la position correspondante dans la séquence de format (soumise en tant que le second argument). Donc le premier item dans la séquence de format remplace de paramètre de positionnement %1, le second item remplace %2, etc. La fonction retourne ce string formaté qui contient les remplacements. Si aucun string n'existe pour un paramètre de positionnement, alors le paramètre de positionnement lui-même est retourné. Cela se produit lorsque l'index d'un paramètre de positionnement est supérieur au nombre d'items dans la séquence de format.

Exemples

- **altova:format-string**('Hello %1, %2, %3', ('Jane','John','Joe')) retourne "Hello Jane, John, Joe"
- **altova:format-string**('Hello %1, %2, %3', ('Jane','John','Joe', 'Tom')) retourne "Hello Jane, John, Joe"
- **altova:format-string**('Hello %1, %2, %4', ('Jane','John','Joe', 'Tom')) retourne "Hello Jane, John, Tom"
- **altova:format-string**('Hello %1, %2, %4', ('Jane','John','Joe')) retourne "Hello Jane, John, %4"

▼ last-chars [altova:]

altova:last-chars(X-Number *as xs:integer*) **asxs:string** XP3.1 XQ3.1

Retourne une chaîne contenant le dernier `X-Number` de caractères de la chaîne obtenue en convertissant la valeur de l'item contextuel en `xs:string`.

Exemples

Si l'item contextuel est **`1234ABCD`** :

- **`altova:last-chars`**`(2)` retourne `CD`
- **`altova:last-chars`**`(5)` retourne `4ABCD`
- **`altova:last-chars`**`(9)` retourne `1234ABCD`

`altova:last-chars(InputString` *`as xs:string`*`, `**`X-Number`** *`as xs:integer`*`)` **`asxs:string`** **XP3.1** **XQ3.1**

Retourne une chaîne contenant le dernier `X-Number` de caractères de la chaîne soumise en tant que l'argument `InputString`.

Exemples

- **`altova:last-chars`**`("2014-01-15", 5)` retourne `01-15`
- **`altova:last-chars`**`("USA", 10)` retourne `USA`

pad-string-left [altova:]

`altova:pad-string-left(StringToPad` *`as xs:string`*`, `**`StringLength`** *`as xs:integer`*`,` **`PadCharacter`** *`as xs:string`*`)` **`asxs:string`** **XP3.1** **XQ3.1**

L'argument `PadCharacter` est un caractère unique. Il est bourré à la gauche de la chaîne pour augmenter le nombre de caractères dans `StringToPad` de manière à ce que ce nombre soit équivalent à la valeur d'entier de l'argument `StringLength`. L'argument `StringLength` peut avoir toute valeur d'entier (positive ou négative), mais le padding n'aura lieu que si la valeur de `StringLength` est supérieure au nombre de caractères dans `StringToPad`. Si `StringToPad`. comporte plus de caractères que la valeur de `StringLength`, alors `StringToPad` ne sera pas modifié.

Exemples

- **`altova:pad-string-left`**`('AP', 1, 'Z')` retourne `'AP'`
- **`altova:pad-string-left`**`('AP', 2, 'Z')` retourne `'AP'`
- **`altova:pad-string-left`**`('AP', 3, 'Z')` retourne `'ZAP'`
- **`altova:pad-string-left`**`('AP', 4, 'Z')` retourne `'ZZAP'`
- **`altova:pad-string-left`**`('AP', -3, 'Z')` retourne `'AP'`
- **`altova:pad-string-left`**`('AP', 3, 'YZ')` retourne une erreur pad-character-too-long

pad-string-right [altova:]

`altova:pad-string-right(StringToPad` *`as xs:string`*`, `**`StringLength`** *`as xs:integer`*`,` **`PadCharacter`** *`as xs:string`*`)` **`asxs:string`** **XP3.1** **XQ3.1**

L'argument `PadCharacter` est un caractère unique. Il est bourré à la droite de la chaîne pour augmenter le nombre de caractères dans `StringToPad` de manière à ce que ce nombre soit équivalent à la valeur d'entier de l'argument `StringLength`. L'argument `StringLength` peut avoir toute valeur d'entier (positive ou négative), mais le padding n'aura lieu que si la valeur de `StringLength` est supérieure au nombre de caractères dans `StringToPad`. Si `StringToPad`. comporte plus de caractères que la valeur de `StringLength`, alors `StringToPad` ne sera pas modifié.

Exemples

- **`altova:pad-string-right`**`('AP', 1, 'Z')` retourne `'AP'`
- **`altova:pad-string-right`**`('AP', 2, 'Z')` retourne `'AP'`

- **altova:pad-string-right**('AP', 3, 'Z') retourne 'APZ'
- **altova:pad-string-right**('AP', 4, 'Z') retourne 'APZZ'
- **altova:pad-string-right**('AP', -3, 'Z') retourne 'AP'
- **altova:pad-string-right**('AP', 3, 'YZ') retourne une erreur pad-character-too-long

repeat-string [altova:]

altova:repeat-string(InputString *as xs:string*, **Repeats** *as xs:integer*) **asxs:string** XP2 XQ1 XP3.1 XQ3.1

Génère une chaîne qui est composée du premier argument `InputString` répété `Repeats` nombre de fois.

Exemples

- **altova:repeat-string**("Altova #", 3) retourne "Altova #Altova #Altova #"

substring-after-last [altova:]

altova:substring-after-last(MainString *as xs:string*, **CheckString** *as xs:string*)
asxs:string XP3.1 XQ3.1

Si `CheckString` est trouvé dans `MainString`, alors la sous-chaîne qui se produit après `CheckString` dans `MainString` est retournée. Si `CheckString` n'est pas trouvé dans `MainString`, la chaîne vide est retournée. Si `CheckString` est une chaîne vide, alors `MainString` est retourné dans sa totalité. S'il y a plus d'une survenance de `CheckString` dans `MainString`, alors la sous-chaîne après la dernière survenance de `CheckString` est retournée.

Exemples

- **altova:substring-after-last**('ABCDEFGH', 'B') retourne 'CDEFGH'
- **altova:substring-after-last**('ABCDEFGH', 'BC') retourne 'DEFGH'
- **altova:substring-after-last**('ABCDEFGH', 'BD') retourne ''
- **altova:substring-after-last**('ABCDEFGH', 'Z') retourne ''
- **altova:substring-after-last**('ABCDEFGH', '') retourne 'ABCDEFGH'
- **altova:substring-after-last**('ABCD-ABCD', 'B') retourne 'CD'
- **altova:substring-after-last**('ABCD-ABCD-ABCD', 'BCD') retourne ''

substring-before-last [altova:]

altova:substring-before-last(MainString *as xs:string*, **CheckString** *as xs:string*)
asxs:string XP3.1 XQ3.1

Si `CheckString` est trouvé dans `MainString`, alors la sous-chaîne qui se produit avant `CheckString` dans `MainString` est retournée. Si `CheckString` n'est pas trouvé dans `MainString`, ou si `CheckString` est une chaîne vide, la chaîne vide est retournée. S'il y a plus d'une survenance de `CheckString` dans `MainString`, alors la sous-chaîne avant la dernière survenance de `CheckString` est retournée.

Exemples

- **altova:substring-before-last**('ABCDEFGH', 'B') retourne 'A'
- **altova:substring-before-last**('ABCDEFGH', 'BC') retourne 'A'
- **altova:substring-before-last**('ABCDEFGH', 'BD') retourne ''
- **altova:substring-before-last**('ABCDEFGH', 'Z') retourne ''
- **altova:substring-before-last**('ABCDEFGH', '') retourne ''
- **altova:substring-before-last**('ABCD-ABCD', 'B') retourne 'ABCD-A'

- **altova:substring-before-last**('ABCD-ABCD-ABCD', 'ABCD') retourne 'ABCD-ABCD-'

substring-pos [altova:]

altova:substring-pos(StringToCheck *as xs:string*, **StringToFind** *as xs:string*)
asxs:integer XP3.1 XQ3.1
Retourne la position de caractère de la première occurrence de `StringToFind` dans la chaîne `StringToCheck`. La position du caractère est retournée en tant qu'un entier. Le premier caractère de `StringToCheck` a la position `1`. Si `StringToFind` ne se produit pas dans le cadre de `StringToCheck`, l'entier `0` est retourné. Pour contrôler la deuxième occurrence ou une occurrence ultérieure de `StringToCheck`, utiliser la signature suivante de cette fonction.

Exemples

- **altova:substring-pos**('Altova', 'to') retourne 3
- **altova:substring-pos**('Altova', 'tov') retourne 3
- **altova:substring-pos**('Altova', 'tv') retourne 0
- **altova:substring-pos**('AltovaAltova', 'to') retourne 3

altova:substring-pos(StringToCheck *as xs:string*, **StringToFind** *as xs:string*, **Integer** *as xs:integer*) **asxs:integer** XP3.1 XQ3.1
Retourne la position de caractère de `StringToFind` dans la chaîne, `StringToCheck`. La recherche de `StringToFind` commence à partir de la position de caractère indiquée par l'argument `Integer` ; la sous-chaîne du caractère avant cette position n'est pas recherchée. Néanmoins, l'entier retourné, est la position de la chaîne trouvée dans le cadre de la chaîne *entière*, `StringToCheck`. Cette signature est utile pour trouver la deuxième position ou une position ultérieure d'une chaîne qui se produit plusieurs fois avec `StringToCheck`. Si `StringToFind` ne se produit pas dans le cadre de `StringToCheck`, l'entier `0` est retourné.

Exemples

- **altova:substring-pos**('Altova', 'to', 1) retourne 3
- **altova:substring-pos**('Altova', 'to', 3) retourne 3
- **altova:substring-pos**('Altova', 'to', 4) retourne 0
- **altova:substring-pos**('Altova-Altova', 'to', 0) retourne 3
- **altova:substring-pos**('Altova-Altova', 'to', 4) retourne 10

trim-string [altova:]

altova:trim-string(InputString *as xs:string*) **asxs:string** XP3.1 XQ3.1
Cette fonction prend un argument `xs:string`, supprime tout espace blanc de tête et de fin et retourne une `xs:string` « nettoyée ».

Exemples

- **altova:trim-string**(" Hello World ") retourne "Hello World"
- **altova:trim-string**("Hello World ") retourne "Hello World"
- **altova:trim-string**(" Hello World") retourne "Hello World"
- **altova:trim-string**("Hello World") retourne "Hello World"
- **altova:trim-string**("Hello World") retourne "Hello World"

trim-string-left [altova:]

altova:trim-string-left(InputString *as xs:string*) **asxs:string** XP3.1 XQ3.1
Cette fonction prend un argument `xs:string`, supprime tout espace blanc de tête, et retourne une `xs:string` nettoyée à gauche.

Exemples

- **altova:trim-string-left**(" Hello World ") retourne "Hello World "
- **altova:trim-string-left**("Hello World ") retourne "Hello World "
- **altova:trim-string-left**(" Hello World") retourne "Hello World"
- **altova:trim-string-left**("Hello World") retourne "Hello World"
- **altova:trim-string-left**("Hello World") retourne "Hello World"

trim-string-right [altova:]

altova:trim-string-right(InputString *as xs:string*) **asxs:string** XP3.1 XQ3.1
Cette fonction prend un argument `xs:string`, supprime tout espace blanc de fin de ligne, et retourne une `xs:string` nettoyée à droite.

Exemples

- **altova:trim-string-right**(" Hello World ") retourne " Hello World"
- **altova:trim-string-right**("Hello World ") retourne "Hello World"
- **altova:trim-string-right**(" Hello World") retourne " Hello World"
- **altova:trim-string-right**("Hello World") retourne "Hello World"
- **altova:trim-string-right**("Hello World") retourne "Hello World"

12.2.2.1.9 Fonctions XPath/XQuery : Divers

L'objectif général suivant des fonctions d'extension XPath/XQuery sont prises en charge dans la version actuelle de MapForce et celles-ci peuvent être utilisées dans (i) des expressions XPath dans un contexte XSLT, ou dans (ii) des expressions XQuery dans un document XQuery.

Note concernant le nommage de fonctions et de l'applicabilité de la langue

Les fonctions d'extension Altova peuvent être utilisées dans les expressions XPath/XQuery. Elles fournissent des fonctions supplémentaires aux fonctions d'ores et déjà disponibles dans la librairie standard des fonctions XPath, XQuery et XSLT. Les fonctions d'extension Altova se trouvent dans **l'espace de nom des fonctions d'extension Altova, `http://www.altova.com/xslt-extensions`**, et sont indiquées dans cette section par le préfixe **`altova:`**, qui est présupposé être lié à cet espace de nom. Veuillez noter que, en ce qui concerne les versions futures de votre produit, la prise en charge d'une fonction peut être interrompue et le comportement de certaines fonctions peut changer. Veuillez consulter la documentation lors des publications à venir pour plus d'informations concernant la prise en charge des fonctions d'extension Altova de cette version.

Fonctions XPath (utilisées dans les expressions XPath dans XSLT) :	XP1 XP2 XP3.1
Fonctions XSLT (utilisées dans les expressions XPath dans	XSLT1 XSLT2 XSLT3

XSLT) :	
Fonctions XQuery (utilisées dans les expressions XQuery dans XQuery) :	XQ1 XQ3.1

decode-string [altova:]

altova:decode-string(Input *as xs:base64Binary*) **as xs:string** XP3.1 XQ3.1
altova:decode-string(Input *as xs:base64Binary*, **Encoding** *as xs:string*) **as xs:string** XP3.1 XQ3.1
Décode l'entrée base64Binary soumise à un string en utilisant l'encodage spécifié. Si aucun codage n'est spécifié, l'encodage UTF-8 est utilisé. Les encodages suivants sont pris en charge: `US-ASCII, ISO-8859-1, UTF-16, UTF-16LE, UTF-16BE, ISO-10646-UCS2, UTF-32, UTF-32LE, UTF-32BE, ISO-10646-UCS4`

Exemples

- **altova:decode-string**(`$XML1/MailData/Meta/b64B`) retourne l'entrée base64Binary en tant qu'un string encodé en UTF-8
- **altova:decode-string**(`$XML1/MailData/Meta/b64B, "UTF-8"`) retourne l'entrée base64Binary en tant qu'un string encodé en UTF-8
- **altova:decode-string**(`$XML1/MailData/Meta/b64B, "ISO-8859-1"`) retourne l'entrée base64Binary en tant qu'un string encodé en ISO-8859-1

.

get-temp-folder [altova:]

altova:get-temp-folder() as `xs:string` XP2 XQ1 XP3.1 XQ3.1
Cette fonction ne prend aucun argument. Elle retourne le chemin vers le dossier temporaire de l'utilisateur actuel.

Exemples

- **altova:get-temp-folder**() retournerait, sur une machine Windows, quelque chose du genre `C:\Users\<UserName>\AppData\Local\Temp\` en tant que `xs:string`.

generate-guid [altova:]

altova:generate-guid() asxs:string XP2 XQ1 XP3.1 XQ3.1
Génère un string GUID unique.

Exemples

- **altova:generate-guid**() retourne (par exemple) `85F971DA-17F3-4E4E-994E-99137873ACCD`

high-res-timer [altova:]

altova:high-res-timer() asxs:double XP3.1 XQ3.1
Retourne une valeur de minuteur haute résolution en secondes. Un minuteur de haute résolution, lorsqu'il est présent dans un système, permet des mesures temporelles de haute précision lorsque celles-ci sont requises (par exemples pour des animations et pour déterminer précisément l'heure d'exécution du code). Cette fonction fournit la résolution du minuteur haute résolution du système.

Exemples

- **altova:high-res-timer**() retourne quelque chose comme '1.16766146154566E6'

parse-html [altova:]

altova:parse-html(HTMLText *as xs:string***) asnode()** XP3.1 XQ3.1
L'argument HTMLText est un string qui contient le texte d'un document HTML. La fonction crée une arborescence HTML depuis le string. Le string soumis peut contenir l'élément HTML ou pas. Dans tous les cas, l'élément racine de l'arborescence est un élément nommé **HTML**. Il est préférable de s'assurer que le code HTML dans le string soumis est un HTML valide.

Exemples

- **altova:parse-html**("<html><head/><body><h1>Header</h1></body></html>") crée une arborescence HTML depuis le string soumis

sleep[altova:]

altova:sleep(Millisecs *as xs:integer***) asempty-sequence()** XP2 XQ1 XP3.1 XQ3.1
Suspend l'exécution de l'opération actuelle pour le nombre de millisecondes donné par l'argument Millisecs.

Exemples

- **altova:sleep**(1000) suspend l'exécution de l'opération actuelle pour 1000 millisecondes.

[Top 567]

12.2.2.2 Fonctions d'extension diverses

Il existe plusieurs types de fonctions prêtes à l'utilisation dans les langages de programmation comme Java et C# qui ne sont pas disponibles en tant que fonctions XQuery/XPath ou en tant que fonctions XSLT. Un bon exemple sont les fonctions mathématiques disponibles en Java, comme sin() et cos(). Si ces fonctions étaient disponibles aux designers des feuilles de style XSLT et des requêtes XQuery, cela augmenterait le champ d'application des feuilles de style et des requêtes et faciliterait considérablement les tâches des créateurs de feuilles de style. Les moteurs XSLT et XQuery utilisés dans un grand nombre de produits Altova prennent en charge l'utilisation des fonctions d'extension dans Java 570 et .NET 579, et pour les scripts MSXSL pour XSLT 585. Cette section décrit comment utiliser les fonctions d'extension et les scripts MSXSL dans vos feuilles de scripts XSLT et les documents XQuery. Les fonctions d'extension disponibles sont organisées dans les sections suivantes :

- Fonctions d'extension Java 570
- Fonctions d'extension .NET 579
- Scripts MSXSL pour XSLT 585

Les deux problèmes principaux considérés pour les descriptions sont : (i) comment sont appelées les fonctions dans les librairies respectives ; et (ii) quelles sont les règles à suivre pour la conversion d'arguments dans un appel de fonction pour obtenir le format d'entrée requis de la fonction, et quelles sont les règles à suivre pour la conversion de retour (résultat de la fonction à l'objet de données XSLT/XQuery).

Exigences

Pour une prise en charge des fonctions d'extension, un Java Runtime Environment (pour l'accès aux fonctions Java) et le cadre de travail .NET Framework 2.0 (minimum, pour l'accès aux fonctions .NET) doit être installé sur la machine qui effectue la transformation XSLT ou l'exécution XQuery, ou doit être accessible pour les transformations.

12.2.2.2.1 Fonctions d'extension Java

Une fonction d'extension Java peut être utilisée dans le cadre d'une expression XPath ou XQuery pour invoquer un constructeur Java ou pour appeler une méthode Java (statique ou d'instance).

Un champ dans une classe Java est considéré être une méthode sans argument. Un champ peut être statique ou d'instance. L'accès aux champs est décrit dans les sous-sections respectives, statiques et d'instance.

Cette section est organisée dans les sous-sections suivantes :

- Java: Constructeurs 575
- Java: Méthodes statiques et champs statiques 576
- Java: Méthodes d'instance et champs d'instance 577
- Types de données : XPath/XQuery en Java 577
- Types de données : Java en XPath/XQuery 579

Veuillez noter

- Si vous utilisez un produit de desktop Altova, les tentatives d'application Altova pour détecter automatiquement le chemin vers la machine virtuelle Java, en lisant (dans cet ordre) : (i) le registre Windows, et (ii) la variable d'environnement **JAVA_HOME**. Vous pouvez aussi ajouter un chemin personnalisé dans le dialogue Options de l'application ; cette entrée prendra la priorité sur tout autre chemin Java VM détecté automatiquement.
- Si vous exécutez un produit serveur Altova sur un appareil Windows, le chemin vers la machine virtuelle sera lu tout d'abord depuis le registre Windows ; si cela échoue, la variable d'environnement **JAVA_HOME** sera utilisée.
- Si vous exécutez un produit de serveur Altova sur un appareil Linux ou macOS, veuillez vous assurer que la variable d'environnement **JAVA_HOME** est définie correctement et que la bibliothèque des appareils virtuels Java (sur Windows, le fichier **jvm.dll)** puisse être située dans le répertoire **\bin\server** ou **\bin\client**.

Forme la fonction d'extension

La fonction d'extension dans l'expression XPath/XQuery doit prendre la forme `prefix:fname()`.

- La partie `prefix:` identifie la fonction d'extension en tant qu'une fonction Java. Elle procède en associant la fonction d'extension avec une déclaration d'espace de nom in-scope, dont l'URI doit commencer avec `java:` (*voir les exemples ci-dessous*). La déclaration d'espace de nom devrait identifier une classe Java, par exemple : `xmlns:myns="java:java.lang.Math"`. Néanmoins, elle pourrait être simplement : `xmlns:myns="java"` (sans les doubles points), l'identification de la classe Java se faisant dans la partie `fname()`de la fonction d'extension.
- La partie `fname()` identifie la méthode Java appelée, et fournit les arguments pour la méthode (*voir les exemples ci-dessous*). Néanmoins, si l'URI d'espace de nom identifié par la partie `prefix:` n'identifie

pas une classe Java (*voir point précédent*), alors la classe Java devrait être identifiée dans la partie `fname()`, avant la classe et séparé de la classe par un point (*voir le second exemple XSLT ci-dessous*).

Note : La classe appelée doit être sur le classpath de la machine.

Exemple XSLT

Voici deux exemples pour démontrer comment appeler une méthode statique. Dans le premier exemple, le nom de classe (`java.lang.Math`) est inclus dans l'URI d'espace de nom et ne doit donc pas être dans la même partie `fname()`. Dans le second exemple, la partie `prefix:` fournit le préfixe `java:` alors que la partie `fname()` identifie la classe et la méthode.

```
<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)" />
```

La méthode nommée dans la fonction d'extension (`cos()` dans l'exemple ci-dessus) doit correspondre au nom d'une méthode statique publique dans la classe Java nommée (`java.lang.Math` dans l'exemple ci-dessus).

Exemple XQuery

Voici un exemple XQuery semblable à l'exemple XSLT ci-dessus:

```
<cosine xmlns:jMath="java:java.lang.Math">
   {jMath:cos(3.14)}
</cosine>
```

Classes Java définies par l'utilisateur

Si vous avez créé vos propres classes Java, les méthodes dans ces classes sont appelées différemment selon que : (i) les classes sont accédées par un fichier JAR ou un fichier classe et (ii) si ces fichiers (JAR ou classe) sont situés dans le répertoire actuel (le même répertoire que le document XSLT ou XQuery) ou pas. Consulter les sections Fichiers de classe définis par l'utilisateur 571 et Fichiers Jar définis par l'utilisateur 574 pour apprendre comment localiser ces fichiers. Veuillez noter que les chemins vers les fichiers de classe ne se trouvant pas dans le répertoire actuel et vers tous les fichiers Jar doivent être spécifiés.

12.2.2.2.1.1 Fichiers de classe définis par l'utilisateur

Si l'accès se produit par un fichier de classe, quatre possibilités s'ouvrent à vous :

- Le fichier de classe se trouve dans un package. Le fichier XSLT ou XQuery se trouve dans le même dossier que le package Java. (*Voir exemple ci-dessous* 572.)
- Le fichier de classe ne se trouve pas dans un package. Le fichier XSLT ou XQuery se trouve dans le même dossier que le fichier de classe. (*Voir exemple ci-dessous* 572.)

- Le fichier de classe se trouve dans un package. Le fichier XSLT ou XQuery se trouve dans un emplacement aléatoire. (*Voir exemple ci-dessous* 573.)
- Le fichier de classe ne se trouve pas dans un package. Le fichier XSLT ou XQuery se trouve dans un emplacement aléatoire. (*Voir exemple ci-dessous* 574.)

Prenons le cas où le fichier de classe ne se trouve pas dans un package et se trouve dans le même dossier que le document XSLT ou XQuery document. Dans ce cas, puisque toutes les classes dans le dossier sont trouvées, l'emplacement du fichier ne doit pas être spécifié. La syntaxe pour identifier une classe est :

```
java:classname
```

alors que

`java:` indique qu'une fonction Java définie par l'utilisateur est appelée ; (les classes Java dans le répertoire actuelles seront chargées par défaut)
`classname` est le nom de la classe de la méthode requise

La classe est identifiée dans un URI d'espace de nom, et l'espace de nom est utilisé pour préfixer un appel de méthode.

Fichier de classe dans un package, le fichier XSLT/XQuery se trouve dans le même dossier que le package Java

L'exemple ci-dessous appelle la méthode `getVehicleType()`de la classe `Car` du package `com.altova.extfunc`. Le package `com.altova.extfunc` se trouve dans le dossier `JavaProject`. Le fichier XSLT se trouve également dans le dossier `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>
```

Fichier de classe est référencé, le fichier XSLT/XQuery se trouve dans le même dossier que le fichier de classe

L'exemple ci-dessous appelle la méthode `getVehicleType()`de la classe `Car`. Partons du principe que (i) le fichier de classe `Car` se trouve dans l'emplacement de dossier suivant : `JavaProject/com/altova/extfunc` et (ii) que ce dossier est le dossier actuel de l'exemple ci-dessous. Le fichier XSLT se trouve aussi dans le dossier `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>
```

Fichier de classe dans un package, le fichier XSLT/XQuery se trouve dans n'importe quel emplacement

L'exemple ci-dessous appelle la méthode `getCarColor()`de la classe `Car` du package `com.altova.extfunc`. Le package `com.altova.extfunc` se trouve dans le dossier `JavaProject`. Le fichier XSLT se trouve dans n'importe quel emplacement. Dans ce cas, l'emplacement du package doit être spécifié dans l'URI en tant que chaîne de requête. La syntaxe est :

```
java:classname[?path=uri-of-package]
```

alors que

`java:` indique qu'une fonction Java définie par l'utilisateur est appelée
`uri-of-package` est l'URI du package Java
`classname` est le nom de la classe de méthode requise

La classe est identifiée dans un URI d'espace de nom et l'espace de nom est utilisé pour préfixer un appel de méthode. L'exemple ci-dessous montre comment accéder à un fichier de classe qui se situe dans un autre répertoire que le répertoire actuel.

```
<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>
```

Fichier de classe référencé, XSLT/XQuery se trouve dans n'importe quel emplacement

L'exemple ci-dessous appelle la méthode `getCarColor()`de la classe `Car`. Si par exemple le fichier de classe `Car` se trouve dans le dossier `C:/JavaProject/com/altova/extfunc` et que le fichier XSLT se trouve dans n'importe quel emplacement. L'emplacement du fichier de classe doit être spécifié dans le cadre de l'URI d'espace de nom en tant que chaîne de requête. La syntaxe est :

```
java:classname[?path==<uri-of-classfile>]
```

alors que

`java:` indique qu'une fonction Java définie par l'utilisateur est appelée
`uri-of-classfile` est l'URI du dossier contenant le fichier de classe
`classname` est le nom de la classe de méthode requise

La classe est identifiée dans un URI d'espace de nom et l'espace de nom est utilisé pour préfixer un appel de méthode. L'exemple ci-dessous montre comment accéder à un fichier de classe qui se situe dans un autre répertoire que le répertoire actuel.

```
<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>
```

Note : Lorsqu'un chemin d'accès est fourni par le biais de la fonction d'extension, le chemin est ajouté au ClassLoader.

12.2.2.2.1.2 Fichiers Jar définis par l'utilisateur

Si l'accès s'effectue par le biais d'un fichier JAR, l'URI du fichier JAR doit être spécifié à l'aide de la syntaxe suivante :

```
xmlns:classNS="java:classname?path=jar:uri-of-jarfile!/"
```

La méthode est ensuite appelée en utilisant le préfixe de l'URI d'espace de nom qui identifie la classe : `classNS:method()`

Dans l'exemple ci-dessus :

`java:` indique qu'une fonction Java est appelée
`classname` est le nom de la classe définie par l'utilisateur
`?` est le séparateur entre le nom de classe et le chemin d'accès
`path=jar:` indique qu'un chemin d'accès vers un fichier JAR est donné
`uri-of-jarfile` est l'URI du fichier jar
`!/` est le délimiteur de fin du chemin d'accès
`classNS:method()` est l'appel de la méthode

En alternative, le nom de classe peut être donné avec l'appel de la méthode. Voici deux exemples de la syntaxe :

```
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()
```

Voici un exemple XSLT complet qui utilise un fichier JAR pour appeler une fonction d'extension 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>
```

Note : Lorsqu'un chemin d'accès est fourni par le biais de la fonction d'extension, le chemin est ajouté au ClassLoader.

12.2.2.2.1.3 Java: Constructeurs

Une fonction d'extension peut être utilisée pour appeler un constructeur Java. Tous les constructeurs sont appelés avec la pseudo-fonction `new()`.

Si le résultat d'un appel de constructeur Java peut être converti implicitement en des types de données XPath/XQuery[579], alors la fonction d'extension Java retournera une séquence qui est un type de données XPath/XQuery. Si le résultat d'un appel de constructeur Java ne peut pas être converti en un type de données XPath/XQuery adéquat, le constructeur crée un objet Java encapsulé avec un type qui est le nom de la classe retournant cet objet Java. Par exemple, si un constructeur pour la classe `java.util.Date` est appelé (`java.util.Date.new()`), alors un objet ayant un type `java.util.Date` est retourné. Le format lexical de

l'objet retourné peut ne pas être conforme au format lexical d'un type de données XPath et la valeur devrait donc être convertie dans le format lexical du type de données XPath requis, puis au type de données XPath requis.

Deux choses peuvent être réalisées avec un objet Java créé par un constructeur :

- Il peut être attribué à une variable :
```
<xsl:variable name="currentdate" select="date:new()"
xmlns:date="java:java.util.Date" />
```
- Il peut être passé à une fonction d'extension (*voir* *Méthode d'instance et Champs d'instance* 577) :
```
<xsl:value-of select="date:toString(date:new())" xmlns:date="java:java.util.Date" />
```

12.2.2.2.1.4 Java: Méthodes statiques et Champs statiques

Une méthode est appelée directement par son nom Java et en fournissant les arguments pour la méthode. Les champs statiques (méthodes qui ne prennent aucun argument), comme les champs de valeur constante `E` et `PI`, sont accédés sans devoir spécifier aucun argument.

Exemples XSLT

Voici quelques exemples montrant comment appeler des méthodes et des champs statiques :

```
<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)" />
```

Veuillez noter que les fonctions d'extension ci-dessus prennent la forme `prefix:fname()`. Le préfixe dans les trois cas indiqués est `jMath:`, qui est associé avec l'URI d'espace de nom `java:java.lang.Math`. (L'URI d'espace de nom doit commencer avec `java:`. Dans les exemples ci-dessus, il est étendu pour contenir le nom de classe (`java.lang.Math`).) La partie `fname()` des fonctions extension doit correspondre au nom d'une classe publique (par ex. `java.lang.Math`) suivi du nom d'une méthode statique publique avec son/ses argument/s (comme `cos(3.14)`) ou un champ statique public (comme `PI()`).

Dans les exemples ci-dessus, Le nom de classe a été inclus dans l'espace de nom URI. S'il n'était pas contenu dans l'URI d'espace de nom, il devrait être inclus dans la partie `fname()` de la fonction d'extension. Par exemple :

```
<xsl:value-of xmlns:java="java:"
              select="java:java.lang.Math.cos(3.14)" />
```

Exemple XQuery

Un exemple semblable dans XQuery serait :

```
<cosine xmlns:jMath="java:java.lang.Math">
```



```
        {jMath:cos(3.14)}
      </cosine>
```

12.2.2.2.1.5 Java: Méthodes d'instance et Champs d'instance

Une méthode d'instance a un objet Java qui lui est passé en tant que son premier argument de l'appel de méthode. Un tel objet Java est généralement créé en utilisant une fonction d'extension (par exemple un appel de constructeur) ou un paramètre/variable de feuille de style. Un exemple XSLT de ce type serait :

```
<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>
```

Dans l'exemple ci-dessus, la valeur du nœud `enrollment/@type` est créée comme suit :

1. Un objet est créé avec un constructeur pour la classe `java.util.Date` (avec le constructeur `date:new()`).
2. Cet objet Java est passé en tant que l'argument de la méthode `jlang.Object.getClass`.
3. L'objet obtenu par la méthode `getClass` est passé en tant que l'argument de la méthode `jlang.Object.toString`.

Le résultat (la valeur de `@type`) aura une chaîne avec la valeur : `java.util.Date`.

Un champ d'instance est théoriquement différent d'une méthode d'instance dans le sens où ce n'est pas un objet Java en soit qui est passé en tant qu'argument au champ d'instance. Au lieu, un paramètre ou une variable est passée en tant que l'argument. Néanmoins, le paramètre/variable peut contenir elle-même la valeur retournée par un objet Java. Par exemple, le paramètre `CurrentDate` prend la valeur retournée par un constructeur pour la classe `java.util.Date`. Cette valeur est ensuite passée en tant qu'un argument à la méthode d'instance `date:toString` afin de fournir la valeur de `/enrollment/@date`.

12.2.2.2.1.6 Types de données : XPath/XQuery en Java

Lorsqu'une fonction Java est appelée depuis l'intérieur d'une expression XPath/XQuery, le type de données des arguments de la fonction est important pour déterminer laquelle des multiples classes Java possédant le même nom est appelé.

dans Java, les règles suivantes sont suivies :

- S'il y a plus d'une seule méthode Java portant le même nom, mais chacune contenant un nombre différent d'arguments que les autres, alors c'est la méthode Java qui correspond le mieux au nombre d'arguments dans l'appel de la fonction qui est sélectionnée.
- La chaîne XPath/XQuery, le nombre et les types de données booléens (*voir la liste ci-dessous*) sont convertis implicitement en un type de données Java correspondant. Si le type XPath/XQuery fourni peut être converti en plus d'un seul type Java (par exemple, `xs:integer`), alors, le type Java sélectionné est celui qui est déclaré pour la méthode sélectionnée. Par exemple, si la méthode Java appelée est `fx(decimal)` et que le type de données XPath/XQuery fourni est `xs:integer`, alors `xs:integer` sera converti dans le type de données `decimal` de Java.

La table ci-dessous recense les conversions implicites de la chaîne XPath/XQuery, le nombre et les types booléens en types de données Java.

`xs:string`	`java.lang.String`
`xs:boolean`	`boolean` (primitive), `java.lang.Boolean`
`xs:integer`	int, long, short, byte, float, double, et les classes de wrapper de ces catégories comme `java.lang.Integer`
`xs:float`	`float` (primitive), `java.lang.Float`, `double` (primitive)
`xs:double`	`double` (primitive), `java.lang.Double`
`xs:decimal`	`float` (primitive), `java.lang.Float`, `double`(primitive), `java.lang.Double`

Les sous-types des types de données du schéma XML recensés ci-dessus (et qui sont utilisés dans XPath et XQuery) seront aussi convertis dans le/s type/s Java correspondant à ce type d'ancêtre du sous-type.

Dans certains cas, il peut ne pas être possible de sélectionner la méthode Java correcte sur la base du cas suivant.

- L'argument fourni est une valeur `xs:untypedAtomic` de 10 et il est prévu pour la méthode `mymethod(float)`.
- Néanmoins, il existe une autre méthode dans la classe qui prend un argument d'un autre type de données : `mymethod(double)`.
- Puisque les noms de la méthode sont les mêmes et que le type fourni (`xs:untypedAtomic`) peut être converti correctement soit à `float` ou `double`, il est possible que `xs:untypedAtomic` soit converti en `double` au lieu de `float`.
- Par conséquence, la méthode sélectionnée ne sera pas la méthode requise et peut ne pas produire le résultat attendu. Pour contourner ce problème, vous pouvez créer une méthode définie par l'utilisateur avec un autre nom et utiliser cette méthode.

Les types qui ne sont pas couverts dans la liste ci-dessus (par exemple, `xs:date`) ne sera pas converti et générera une erreur. Néanmoins, veuillez noter que dans certains cas, il peut être possible de créer le type Java requis en utilisant un constructeur Java.

12.2.2.2.1.7 Types de données : Java en XPath/XQuery

Lorsqu'une méthode Java retourne une valeur, le type de donnée de la valeur est de type chaîne, numérique ou booléen, alors il est converti dans le type XPath/XQuery correspondant. Par exemple, les types de données `java.lang.Boolean` et `boolean` de Java sont convertis en `xsd:boolean`.

Des arrays unidimensionnels retournés par des fonctions sont étendus à une séquence. Les arrays multidimensionnels ne seront pas convertis et devraient donc être encapsulés.

Lorsqu'un objet Java encapsulé ou un type de données différent de chaîne, numérique ou booléen est retourné, vous pouvez assurer la conversion dans le type XPath/XQuery requis tout d'abord en utilisant une méthode Java (par ex.`toString`) pour convertir l'objet Java en une chaîne. Dans XPath/XQuery, la chaîne peut être modifiée pour correspondre à la représentation lexicale du type requis puis convertie dans le type requis (par exemple, en utilisant l'expression `cast as`).

12.2.2.2.2 Fonctions d'extension .NET

Si vous travaillez sur la plate-forme .NET d'une machine Windows, vous pouvez utiliser les fonctions d'extension écrites dans un des langages .NET (par exemple, C#). Une fonction d'extension .NET peut être utilisée dans une expression XPath ou XQuery pour invoquer un constructeur, une propriété ou une méthode (statique ou instance) dans une classe .NET.

Une propriété d'une classe .NET est appelée en utilisant la syntaxe `get_PropertyName()`.

Cette section est organisée dans les sous-sections suivantes :

Forme de la fonction d'extension

La fonction d'extension de l'expression XPath/XQuery doit avoir la forme `prefix:fname()`.

- La partie `prefix:` est associée avec un URI qui identifie la classe .NET en cours d'adressage.
- La partie `fname()` identifie le constructeur, la propriété ou la méthode (statique ou d'instance) dans la classe .NET, et fournit un/des argument/s, le cas échéant.
- L'URI doit commencer avec `clitype:` (qui identifie la fonction comme étant une fonction d'extension .NET).
- La forme `prefix:fname()` de la fonction d'extension peut être utilisée avec les classes de système et avec les classes dans un assemblage chargé. Néanmoins, si une classe doit être chargée, les paramètres supplémentaires contenant l'information requise devront être fournis.

Paramètres

Pour charger un assemblage, utiliser les paramètres suivants :

`asm`	Le nom de l'assemblage à charger.
`ver`	Le numéro de la version (quatre entiers maximum séparés par des points).
`sn`	Le jeton clé du nom solide de l'assemblage (16 chiffres hex).
`from`	Un URI qui donne l'emplacement de l'assemblage (DLL) à charger. Si l'URI est relatif, il est relatif au document XSLT ou XQuery. Si ce paramètre est présent, tout autre paramètre est ignoré.
`partialname`	Le nom partiel de l'assemblage. Il est fourni dans `Assembly.LoadWith.PartialName()`, qui tentera de charger l'assemblage. Si `partialname` est présent, tout autre paramètre est ignoré.
`loc`	La région, par exemple, `en-US`. Le défaut est `neutral`.

Si l'assemblage doit être chargé depuis un DLL, utiliser le paramètre `from` et omettre le paramètre `sn`. Si l'assemblage doit être chargé depuis le Global Assembly Cache (GAC), utiliser le paramètre `sn` et omettre le paramètre `from`.

Un signe d'interrogation doit être inséré avant le premier paramètre, et les paramètres doivent être séparés par un point-virgule. Le nom du paramètre donne sa valeur avec un signe égal (*voir exemple ci-dessous*).

Exemples de déclarations d'espace de nom

Un exemple de déclaration d'espace de nom dans XSLT qui identifie le système class `System.Environment`:

```
xmlns:myns="clitype:System.Environment"
```

Un exemple d'une déclaration d'espace de nom dans XSLT qui identifie la classe devant être chargée en tant que `Trade.Forward.Scrip`:

```
xmlns:myns="clitype:Trade.Forward.Scrip?asm=forward;version=10.6.2.1"
```

Un exemple d'une déclaration d'espace de nom dans XQuery qui identifie la classe de système `MyManagedDLL.testClass`:. Deux cas peuvent être distingués :

1. Lorsque l'assemblage est chargé depuis le GAC:
```
declare namespace cs="clitype:MyManagedDLL.testClass?asm=MyManagedDLL;
ver=1.2.3.4;loc=neutral;sn=b9f091b72dccfba8";
```

2. Lorsque l'assemblage est chargé depuis le DLL (références partielles ou complète ci-dessous) :
```
declare namespace cs="clitype:MyManagedDLL.testClass?from=file:///C:/Altova
Projects/extFunctions/MyManagedDLL.dll;

declare namespace cs="clitype:MyManagedDLL.testClass?from=MyManagedDLL.dll;
```

Exemple XSLT

Voici un exemple complet XSLT qui appelle les fonctions dans la classe de système `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 déclaration d'espace de nom de l'élément `math` associe le préfixe `math:` avec l'URI `clitype:System.Math`. Le début `clitype:` de l'URI indique que ce qui suit identifie soit une classe de système soit une classe chargée. Le préfixe `math:` dans les expressions XPath associe les fonctions d'extension avec l'URI (et, par extension, la classe) `System.Math`. Les fonctions d'extension identifient des méthodes dans la classe `System.Math` et fournissent les arguments le cas échéant.

Exemple XQuery

Voici un fragment d'exemple XQuery similaire à l'exemple XSLT ci-dessus :

```
<math xmlns:math="clitype:System.Math">
   {math:Sqrt(9)}
</math>
```

Comme pour l'exemple XSLT ci-dessus, la déclaration d'espace de nom identifie la classe .NET, dans ce cas une classe de système. L'expression XQuery identifie la méthode à appeler et fournit l'argument.

12.2.2.2.2.1 .NET: Constructeurs

Une fonction d'extension peut être utilisée pour appeler un constructeur .NET. Tous les constructeurs sont appelés avec la pseudo-fonction `new()`. S'il y a plus d'un constructeur pour une classe, le constructeur qui correspond le mieux au nombre d'arguments fourni est sélectionné. Si aucun constructeur n'est jugé conforme aux arguments fournis, alors une erreur `'No constructor found'` sera retournée.

Des constructeurs qui retournent des types de données XPath/XQuery

Si le résultat d'un appel de constructeur .NET peut être converti implicitement dans des types de données XPath/XQuery [579], alors la fonction d'extension .NET retournera une séquence qui est un type de données XPath/XQuery.

Des constructeurs qui retournent des objets .NET

Si le résultat d'un appel de constructeur .NET ne peut pas être converti en un type de données XPath/XQuery adéquat, le constructeur crée un objet .NET encapsulé avec un type qui est le nom de la classe retournant cet objet. Par exemple, si un constructeur pour la classe `System.DateTime` est appelé (avec `System.DateTime.new()`), alors un objet ayant le type `System.DateTime` est retourné.

Le format lexical de l'objet retourné peut ne pas être conforme au format lexical d'un type de données XPath. Dans ce cas, la valeur retournée devrait donc être : (i) convertie dans le format lexical du type de données XPath requis ; puis (ii) au type de données XPath requis.

Trois choses peuvent être réalisées avec un objet .NET créé par un constructeur :

- Il peut être réalisé dans une variable :
  ```
  <xsl:variable name="currentdate" select="date:new(2008, 4, 29)"
  xmlns:date="clitype:System.DateTime" />
  ```
- Il peut être passé à une fonction d'extension (*voir Méthode d'instance et Champs d'instance* [577]):
  ```
  <xsl:value-of select="date:ToString(date:new(2008, 4, 29))"
  xmlns:date="clitype:System.DateTime" />
  ```
- Il peut être converti en une chaîne, nombre ou booléenne :
- ```
  <xsl:value-of select="xs:integer(date:get_Month(date:new(2008, 4, 29)))"
  xmlns:date="clitype:System.DateTime" />
  ```

12.2.2.2.2.2 .NET: Méthodes statiques et Champs statiques

Une méthode est appelée directement par son nom et en fournissant les arguments pour la méthode. Le nom utilisé dans l'appel doit être exactement conforme à une méthode statique publique dans la classe spécifiée. Si le nom de la méthode et le nombre des arguments qui ont été donnés dans l'appel de la fonction correspondent à plus d'une méthode dans une classe, alors les types des arguments fournis sont évalués pour obtenir la meilleure correspondance possible. Si une correspondance ne peut pas être trouvée sans ambiguïté, une erreur sera rapportée.

Note : Un champ dans une classe .NET est considéré être une méthode sans argument. Une propriété est appelée en utilisant la syntaxe `get_PropertyName()`.

Exemples

Un exemple XSLT montrant un appel vers une méthode avec un argument (`System.Math.Sin(arg)`):

```
<xsl:value-of select="math:Sin(30)" xmlns:math="clitype:System.Math"/>
```
```

Un exemple XSLT montrant un appel vers un champ (considéré une méthode sans argument) (`System.Double.MaxValue()`):

```
<xsl:value-of select="double:MaxValue()" xmlns:double="clitype:System.Double"/>
```

Un exemple XSLT montrant un appel vers une propriété (syntaxe est `get_PropertyName()`) (`System.String()`):

```
<xsl:value-of select="string:get_Length('my string')"
xmlns:string="clitype:System.String"/>
```

Un exemple XQuery montrant un appel vers une méthode avec un argument (`System.Math.Sin(arg)`):

```
<sin xmlns:math="clitype:System.Math">
   { math:Sin(30) }
</sin>
```

12.2.2.2.2.3 .NET: Méthodes d'instance et Champs d'instance

Une méthode d'instance a un objet .NET qui lui est passé en tant que son premier argument de l'appel de méthode. Un tel objet .NET est généralement créé en utilisant une fonction d'extension (par exemple un appel de constructeur) ou un paramètre/variable de feuille de style. Un exemple XSLT de ce type serait :

```
<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>
```

Dans l'exemple ci-dessus, un constructeur `System.DateTime` (`new(2008, 4, 29)`) est utilisé pour créer un objet .NET de type `System.DateTime`. Cet objet est créé deux fois, une fois en tant que la valeur de la variable `releasedate`, une deuxième fois en tant que le premier et seul argument de la méthode

`System.DateTime.ToString()`. La méthode d'instance `System.DateTime.ToString()` est appelée deux fois, les deux fois avec le constructeur `System.DateTime` (`new(2008, 4, 29)`) en tant que son premier et seul argument. Dans une de ces instances, la variable `releasedate` est utilisée pour obtenir l'objet .NET.

Méthodes d'instance et champs d'instance

La différence entre une méthode d'instance et un champ d'instance est théorique. Dans une méthode d'instance, un objet .NET est passé directement en tant qu'un argument ; dans un champ d'instance, un paramètre ou une variable est passé à la place—bien que le paramètre ou la variable puisse contenir un objet .NET. Par exemple, dans l'exemple ci-dessus, la variable `releasedate` contient un objet .NET, et c'est cette variable qui est passée en tant que l'argument de `ToString()` dans le deuxième constructeur d'élément `date`. C'est pourquoi, l'instance `ToString()` dans le premier élément `date` est une méthode d'instance alors que le deuxième est considéré être un champ d'instance. Le résultat produit dans les deux instances, néanmoins, est le même.

12.2.2.2.2.4 Types de données : XPath/XQuery en .NET

Lorsqu'une fonction d'extension .NET est utilisée dans une expression XPath/XQuery, les types de données des arguments de la fonction sont importants pour déterminer laquelle des méthodes .NET multiples possédant le même nom est appelée.

Dans .NET, les règles suivantes sont suivies :

- S'il y a plus d'une méthode portant le même nom dans une classe, les méthodes disponibles pour la sélection sont réduites à celles qui ont le même nombre d'arguments que l'appel de fonction.
- Les types de données chaîne, nombre et booléen XPath/XQuery (*voir la liste ci-dessous*) sont convertis implicitement en un type de données .NET correspondant. Si le type XPath/XQuery fourni peut être converti en plus d'un type .NET (par exemple, `xs:integer`), puis, ce type .NET est sélectionné ce qui est déclaré pour la méthode sélectionnée. Par exemple, si la méthode .NET appelée est `fx(double)` et que le type de données fourni XPath/XQuery est `xs:integer`, alors `xs:integer` sera converti dans le type de données `double` de .NET.

La table ci-dessous recense les conversions implicites des types chaîne, nombre et booléen XPath/XQuery en types de données .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`

Les sous-types des types de données de XML Schema recensés ci-dessus (et qui sont utilisés dans XPath et Query) seront aussi convertis dans le/s type/s .NET correspondant au type ancêtre du sous-type.

Dans certains cas, il peut ne pas être possible de sélectionner la méthode .NET correcte sur la base des informations fournies. Veuillez considérer le cas suivant, par exemple.

- L'argument fourni est une valeur `xs:untypedAtomic` de 10 et est prévue pour la méthode `mymethod(float)`.
- Néanmoins, il y a une autre méthode dans la classe qui prend un argument d'un autre type de données : `mymethod(double)`.
- Puisque les noms de la méthode et le type fourni (`xs:untypedAtomic`) ont pu être convertis correctement en `float` ou `double`, il est possible que `xs:untypedAtomic` soit converti en `double` au lieu de `float`.
- Par conséquence, la méthode sélectionnée ne sera pas celle requise et peut ne pas produire le résultat attendu. Pour contourner ce problème, vous pouvez créer une méthode définie par l'utilisateur avec un nom différent et utiliser cette méthode.

Les types qui ne sont pas couverts dans la liste ci-dessus (par exemple `xs:date`) ne seront pas convertis et généreront une erreur.

12.2.2.2.2.5 Types de données : .NET en XPath/XQuery

Lorsqu'une méthode .NET retourne une valeur et que le type de données de la valeur est de type chaîne, numérique ou booléenne, il est converti dans le type XPath/XQuery correspondant. Par exemple, le type de données `decimal` de .NET est converti en `xsd:decimal`.

Lorsqu'un objet .NET ou un type de données différent de chaîne, numérique ou booléenne est retourné, vous pouvez assurer la conversion dans le type XPath/XQuery requis tout d'abord en utilisant une méthode .NET (par ex. `System.DateTime.ToString()`) pour convertir l'objet .NET en une chaîne. Dans XPath/XQuery, la chaîne peut être modifiée pour correspondre à la représentation lexicale du type requis puis convertie dans le type requis (par exemple, en utilisant l'expression `cast as`).

12.2.2.2.3 Scripts MSXSL pour XSLT

L'élément `<msxsl:script>` contient des fonctions définies par l'utilisateur et des variables qui peuvent être appelées depuis des expressions XPath dans la feuille de style XSLT. Le `<msxsl:script>` et un élément de niveau supérieur, c'est à dire, qu'il doit être un élément enfant de `<xsl:stylesheet>` ou `<xsl:transform>`.

L'élément `<msxsl:script>` doit être dans l'espace de nom `urn:schemas-microsoft-com:xslt` (*voir exemple ci-dessus*).

Langage de script et espace de nom

Le langage de script utilisé dans le bloc est spécifié dans l'attribut `language` de l'élément `<msxsl:script>` et l'espace de nom à utiliser pour les appels de fonction depuis les expressions XPath est identifié avec l'attribut `implements-prefix` (*voir ci-dessous*).

```
<msxsl:script language="scripting-language" implements-prefix="user-namespace-prefix">
```

```
   function-1 or variable-1
   ...
   function-n or variable-n

</msxsl:script>
```

L'élément `<msxsl:script>` interagit avec le Windows Scripting Runtime, donc seuls des langages installés sur votre machine peuvent être utilisés dans l'élément `<msxsl:script>`. **La plate-forme .NET Framework 2.0 ou plus récente doit être installée pour pouvoir utiliser les scripts MSXSL.** Par conséquent, les langages de script .NET peuvent être utilisés dans l'élément `<msxsl:script>`.

L'attribut de langage doit accepter les mêmes valeurs que l'attribut `language` dans l'élément HTML `<script>`. Si l'attribut de langage n'est pas spécifié, alors Microsoft JScript est assumé par défaut.

L'attribut `implements-prefix` prend une valeur qui est un préfixe d'un nom d'espace in-scope déclaré. Cet espace de nom sera généralement un espace de nom d'utilisateur qui a été réservé pour une librairie de fonction. Toutes les fonctions et les variables définies dans l'élément `<msxsl:script>` se trouveront dans l'espace de nom identifié par le préfixe spécifié dans l'attribut `implements-prefix`. Lorsqu'une fonction est appelée depuis une expression XPath, le nom de la fonction entièrement qualifié doit se trouver dans le même espace de nom que la définition de la fonction.

Exemple

Voici un exemple d'une feuille de stype XSLT complète qui utilise une fonction définie dans un élément `<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>
```

Types de données

Les valeurs des paramètres passés dans et hors du bloc de script sont limitées aux types de données XPath. Cette restriction ne s'applique pas aux données passées parmi les fonctions et les variables dans le bloc du script.

Assemblages

Un assemblage peut être importé dans le script en utilisant l'élément **msxsl:assembly**. L'assemblage est identifié par le biais d'un nom ou d'un URI. L'assemblage est importé lors de la compilation de la feuille de style. Voici une simple représentation de la manière d'utiliser l'élément `msxsl:assembly`.

```
<msxsl:script>
   <msxsl:assembly name="myAssembly.assemblyName" />
   <msxsl:assembly href="pathToAssembly" />

   ...

</msxsl:script>
```

Le nom d'assemblage peut être un nom complet comme :

```
"system.Math, Version=3.1.4500.1 Culture=neutral PublicKeyToken=a46b3f648229c514"
```

ou un nom court comme `"myAssembly.Draw"`.

Espaces de nom

Les espaces de nom peuvent être déclarés avec l'élément **msxsl:using**. Cela permet aux classes d'assemblage d'être écrits dans le script sans leurs espaces de nom, ce qui vous épargne un gros travail de saisie. Voici comment l'élément `msxsl:using` est utilisé de manière à déclarer des espaces de nom.

```
<msxsl:script>
   <msxsl:using namespace="myAssemblyNS.NamespaceName" />

   ...

</msxsl:script>
```

La valeur de l'attribut `namespace` est le nom de l'espace de nom.

12.3 Données techniques

Cette section contient des informations utiles concernant certains aspects techniques de votre logiciel. Cette information est organisée dans les sections suivantes :

12.3.1 SE et exigences de mémoire

Système d'exploitation

Les applications logicielles d'Altova sont disponibles pour les plateformes suivantes :

- Windows 10, Windows 11
- Windows Server 2016 ou plus récent

Mémoire

Étant donné que le logiciel est rédigé en C++, il ne nécessite pas la performance d'un Java Runtime Environment et nécessite généralement moins de mémoire que les applications basées sur Java comparables. Néanmoins, chaque document est chargé entièrement dans la mémoire de manière à ce qu'il puisse être parsé complètement et pour améliorer la vitesse de la consultation et de l'édition. Les exigences de mémoire augmentent avec la taille du document.

Les exigences de mémoire sont aussi affectées par un historique de la fonction Annuler non limité. Si vous coupez/collez sans arrêt de larges sélections dans des documents volumineux, la mémoire disponible peut baisser rapidement.

12.3.2 Moteurs Altova

Validateur XML

Lors de l'ouverture d'un document XML, l'application utilise son validateur XML intégré pour vérifier la bonne formation, pour valider le document par rapport à un schéma (si spécifié), et de générer des arborescences et infosets. Le validateur XML est aussi utilisé pour fournir une édition intelligente pendant que vous éditez des documents et pour afficher dynamiquement toute erreur de validation qui peut se produire.

Le validateur XML intégré met en place les spécifications de la Final Recommendation of the W3C's XML Schema 1.0 et 1.1. Les nouveaux développements recommandés par le Groupe de travail de Schéma XML de W3C sont incorporés en continu dans le validateur XML, afin que les produits Altova vous donnent un environnement de développement de pointe.

Moteurs XSLT et XQuery

Les produits Altova utilisent les moteurs Altova XSLT 1.0, 2.0 et 3.0, et les moteurs Altova XQuery 1.0 et 3.1. Si un de ces moteurs est inclus dans le produit, la documentation concernant le comportement spécifique à la mise en place pour chaque moteur est indiquée dans les annexes de la documentation.

Note: Altova MapForce génère du code utilisant les moteurs XSLT 1.0, 2.0 et XQuery 1.0.

12.3.3 Prise en charge Unicode

Les produits XML d'Altova propose une prise en charge complète d'Unicode. Pour éditer un document XML, vous devrez utiliser une police d'écriture qui sera également prise en charge par ce document.

Veuillez noter que la plupart des polices ne contient qu'un sous-ensemble très spécifique de la plage Unicode et qu'elles sont donc généralement taillées à la mesure du système d'écriture correspondant. Si vous voyez apparaître un texte déformé, une des raisons peut être que la police que vous avez sélectionnée ne contient pas les symboles exigés. Il est donc utile de disposer d'une police qui couvre toute la plage Unicode, surtout lors de l'édition de documents XML dans des langues ou des systèmes d'écriture variés. Une police Unicode typique utilisée sur les PC Windows est Arial Unicode MS.

Dans le dossier `/Examples` de votre dossier d'application, vous trouverez un fichier XHTML appelé `UnicodeUTF-8.html` qui contient la phrase suivante dans plusieurs langues et systèmes d'écriture :

- *When the world wants to talk, it speaks Unicode*
- *Quand le monde veut communiquer, il parle en Unicode*
- *Wenn die Welt miteinander spricht, spricht sie Unicode*
- 世界的に話すなら、Unicode です。

Ouvrez ce fichier XHTML pour voir un aperçu des possibilités d'Unicode et pour indiquer que les systèmes d'écriture sont pris en charge par les polices disponibles sur votre PC.

12.3.4 Utilisation Internet

Les applications Altova initieront des connexions Internet pour vous dans les situations suivantes :

- Si vous cliquez sur "Demander code-clé d'évaluation" dans le dialogue d'Enregistrement (**Aide | Activation du logiciel**), les trois champs dans le dialogue d'enregistrement seront transférés vers notre serveur web au moyen d'une connexion http (port 80) normale et le code-clé d'évaluation sera renvoyé au client par le biais d'un e-mail SMTP normal.
- Dans certains produits Altova, vous pouvez ouvrir un fichier dans Internet (**Fichier | Ouvrir | Passer à URL**). Dans ce cas, le document est extrait à l'aide d'une des méthodes de protocole et connexions suivantes : HTTP (normalement port 80), FTP (normalement port 20/21), HTTPS (normalement port 443). Vous pouvez aussi exécuter un serveur HTTP sur le port 8080. (Dans le dialogue URL, spécifier le port après le nom de serveur et un double point.)
- Si vous ouvrez un document XML qui réfère à un Schéma XML ou à un DTD et que le document est spécifié par une URL, le document de schéma référencé est aussi extrait par le biais d'une connexion HTTP (port 80) ou un autre protocole spécifié dans l'URL (voir Point 2 ci-dessus). Un document de

schéma sera aussi extrait lorsqu'un fichier XML est validé. Veuillez noter que la validation peut avoir lieu automatiquement lors de l'ouverture d'un document si vous avez instruit l'application de procéder de cette manière (dans l'onglet Fichier du dialogue Options (**Outils | Options**)).

- Dans les applications Altova utilisant WSDL et SOAP, les connexions de service web sont définies par les documents WSDL.
- Si vous utilisez la commande **Envoyer par courrier électronique** (**Fichier | Envoyer par courrier électronique**) dans XMLSpy, la sélection ou le fichier actuel est envoyé avec tout programme d'e-mail conforme à MAPI et installé sur le PC de l'utilisateur.
- Fait partie de l'Activation du logiciel et du LiveUpdate tel que décrit ultérieurement dans l'Accord de licence de logiciel Altova.

12.4 Informations de licence

Cette section contient des informations concernant :

- la distribution de ce logiciel
- l'activation de logiciel et le license metering
- le contrat de licence régissant l'usage de ce logiciel

Veuillez lire ces informations attentivement. Elles ont force obligatoire puisque vous avez accepté ces termes lors de l'installation de ce logiciel.

Pour consulter les termes de toute licence Altova, rendez-vous sur la page des informations juridiques Altova sur le site web Altova.

12.4.1 Distribution électronique de logiciel

Ce produit est disponible par le biais de la distribution électronique de logiciel, une méthode de distribution qui fournit les avantages uniques suivants :

- Vous pouvez évaluer gratuitement le logiciel pendant 30 jours avant de vous décider à l'achat. (*Note : Altova MobileTogether Designer dispose d'une une licence gratuite.*)
- Une fois que vous avez décidé d'acheter le logiciel, vous pouvez passer vos commandes en ligne sur le site web Altova et vous obtiendrez en quelques minutes un produit bénéficiant d'une pleine licence.
- Lorsque vous passez une commande en ligne, vous disposerez toujours de la dernière version de nos logiciels.
- Le pack de produits comprend une aide sur écran qui peut être accédé depuis l'intérieur de l'interface de l'application. La dernière version du manuel d'utilisateur est disponible sous www.altova.com (i) sous format HTML pour une navigation en ligne, et (ii) sous format PDF pour le téléchargement (et pour imprimer si vous préférez avoir recours à une documentation en papier).

Période d'évaluation de 30 jours

Après avoir téléchargé le produit, vous pourrez évaluer celui-ci gratuitement pour une période de jusqu'à 30 jours. Au bout d'environ 20 jours de cette période d'évaluation, le logiciel commencera à vous rappeler qu'il n'est pas encore sous licence. Le message de rappel s'affichera une fois à chaque fois que vous démarrerez l'application. Si vous souhaitez continuer à utiliser le programme à l'issue de la période d'évaluation de 30 jours, vous devrez acheter une licence de produit, qui est fournie sous la forme d'un fichier de licence contenant un code-clé. Déverrouiller le produit en chargeant le fichier de licence dans le dialogue d'activation du logiciel de votre produit.

Vous pouvez acheter des licences de produit dans la boutique en ligne du https://shop.altova.com/.

Transmettre le logiciel à d'autres collaborateurs dans votre entreprise à des fins d'évaluation

Si vous souhaitez distribuer la version d'évaluation dans le cadre de votre réseau d'entreprise, ou si vous prévoyez de l'utiliser sur un PC qui n'est pas connecté à Internet, vous pourrez uniquement distribuer le fichier d'installation, à condition qu'il ne soit pas modifié de quelque manière que ce soit. Toute personne accédant au programme d'installation du logiciel que vous avez fourni doit demander son propre code-clé d'évaluation de 30

jours et devra aussi acheter une licence à l'issue de la période d'évaluation afin de pouvoir continuer à utiliser le produit.

12.4.2 Activation de logiciel et le license metering

En tant que partie intégrante de l'Activation du logiciel Altova, le logiciel peut utiliser votre réseau interne et votre connexion Internet à des fins de transmission des données relatives à la licence au moment de l'installation, de l'enregistrement, de l'utilisation ou de la mise à jour d'un serveur de licence utilisé par Altova et valider l'authenticité des données relatives à la licence pour protéger Altova contre une utilisation sans licence ou illégale du logiciel et pour améliorer le service clientèle. L'activation est basée sur l'échange des données relatives aux licences comme les systèmes d'exploitation, l'adresse IP, la date/heure, la version de logiciel et le nom de l'ordinateur, ainsi que d'autres informations échangées entre votre ordinateur et un serveur de licence Altova.

Votre produit Altova comporte un module intégré de contrôle des licences qui vous aide à éviter toute violation non-intentionnelle du contrat de licence de l'utilisateur final. Votre produit est licencé soit en tant qu'une installation utilisateur simple soit en tant qu'installation multi-utilisateur, et le module de contrôle des licences permet de vous assurer qu'aucune licence outre celles accordées pour le nombre d'utilisateurs sous licence n'utilise l'application simultanément.

Cette technologie de contrôle des licences utilise votre réseau local (LAN) pour communiquer entre les instances de l'application exécutée sur plusieurs ordinateurs.

Licence simple

Lorsque l'application est démarrée dans le cadre du processus de contrôle de la licence, le logiciel envoie un bref datagramme de diffusion pour trouver d'autres instances du produit exécuté sur d'autres ordinateurs dans le même segment de réseau. S'il n'obtient pas de réponses, il ouvrira un port pour écouter d'autres instances de l'application.

Licence utilisateurs multiples

Si plus d'une seule instance de l'application est utilisée dans le même LAN, ces instances communiqueront brièvement l'une avec l'autre lors du démarrage. Ces instances échangent des codes-clés afin de vous aider à mieux déterminer que le nombre de licences concurrentes achetées n'est pas violé accidentellement. Il s'agit de la même technologie de contrôle des licences généralement utilisée dans l'univers Unix et dans un certain nombre d'outils de développement de bases de données. Elle permet aux clients Altova d'acheter des licences multi-utilisateurs d'utilisation simultanée à des prix raisonnables.

Nous avons également conçu les applications de manière à ce qu'elles envoient des paquets de réseau peu importants et peu nombreux pour ne pas surcharger votre réseau. Les ports TCP/IP (2799) utilisés par votre produit Altova sont officiellement enregistrés auprès de l'IANA (*voir IANA Service Name Registry pour plus de détails*) et notre module le contrôle de licence est testé et éprouvé technologiquement.

Si vous utilisez un pare-feu, vous pourrez éventuellement apercevoir des communications sur le port 2799 entre les ordinateurs qui exécutent les produits Altova. Vous pouvez, bien évidemment, bloquer ce trafic entre les groupes différents dans votre entreprise, du moment que vous pouvez assurer par d'autres moyens que votre contrat de licence n'a pas été violé.

Vous noterez également que, si vous êtes en ligne, votre logiciel Altova contient de nombreuses fonctions utiles ; celles-ci ne concernent pas la technologie de contrôle des licences.

Note à propos des certificats

Votre application Altova contacte le serveur de mise sous licence Altova (`link.altova.com`) via HTTPS. Pour établir cette communication, Altova utilise un certificat SSL enregistré. Si ce certificat est remplacé (par exemple, par votre département IT ou une agence externe), votre application Altova vous avertira que la connexion n'est pas sûre. Vous pourriez utiliser le certificat de remplacement pour lancer votre application Altova, mais vous le ferez à vos propres risques et périls. Si vous voyez un message d'avertissement *Connexion non-sécurisée*, vérifiez l'origine du certificat et consultez votre équipe IT (qui sera en mesure de décider si l'interception et le remplacement du certificat Altova devrait continuer ou pas).

Si votre organisation nécessite d'utiliser son propre certificat (par exemple, pour surveiller la communication de et vers les machines client), nous vous recommandons d'installer le logiciel de gestion de licence gratuit d'Altova, Altova LicenseServer, dans votre réseau. Sous cette configuration, les appareils de client peuvent continuer d'utiliser les certificats de votre organisation, alors que l'Altova LicenseServer peut être autorisé à utiliser le certificat Altova pour une communication avec Altova.

12.4.3 Altova Contrat de licence de l'utilisateur final

- Le contrat Altova de licence de l'utilisateur final est disponible ici : https://www.altova.com/fr/legal/eula
- La politique de confidentialité d'Altova est disponible ici : https://www.altova.com/fr/privacy

Index

A

B

C

D

E

F

G

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z